MAP-BASED LOCALIZATION FOR UNMANNED AERIAL VEHICLE NAVIGATION

JULIEN LI-CHEE-MING

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN EARTH AND SPACE SCIENCE

YORK UNIVERSITY

TORONTO, ONTARIO

JULY 2017

ABSTRACT

Unmanned Aerial Vehicles (UAVs) require precise pose estimation when navigating in indoor and GNSS-denied / GNSS-degraded outdoor environments. The possibility of crashing in these environments is high, as spaces are confined, with many moving obstacles. There are many solutions for localization in GNSS-denied environments, and many different technologies are used. Common solutions involve setting up or using existing infrastructure, such as beacons, Wi-Fi, or surveyed targets. These solutions were avoided because the cost should be proportional to the number of users, not the coverage area. Heavy and expensive sensors, for example a high-end IMU, were also avoided. Given these requirements, a camera-based localization solution was selected for the sensor pose estimation. Several camera-based localization approaches were investigated. Map-based localization methods were shown to be the most efficient because they close loops using a pre-existing map, thus the amount of data and the amount of time spent collecting data are reduced as there is no need to re-observe the same areas multiple times. This dissertation proposes a solution to address the task of fully localizing a monocular camera onboard a UAV with respect to a known environment (i.e., it is assumed that a 3D model of the environment is available) for the purpose of navigation for UAVs in structured environments.

Incremental map-based localization involves tracking a map through an image sequence. When the map is a 3D model, this task is referred to as model-based tracking. A by-product of the tracker is the relative 3D pose (position and orientation) between the camera and the object being tracked. State-of-the-art solutions advocate that tracking geometry is more robust than tracking image texture because edges are more invariant to changes in object appearance and lighting. However, model-based trackers have been limited to tracking small simple objects in small environments. An assessment was performed in tracking larger, more complex building models, in larger environments. A state-of-the art model-based tracker called ViSP (Visual Servoing Platform) was applied in tracking outdoor and indoor buildings using a UAV's low-cost camera. The assessment revealed weaknesses at large scales. Specifically, ViSP failed when tracking was lost, and needed to be manually re-initialized. Failure occurred when there was a lack of model features in the camera's field of view, and because of rapid camera motion. Experiments revealed that ViSP achieved positional accuracies similar to single point positioning solutions obtained from single-frequency (L1) GPS observations – standard deviations around 10 metres. These errors were considered to be large, considering the geometric accuracy of the 3D model used in the experiments was 10 to 40 cm. The first contribution of this dissertation proposes to increase the performance of the localization system by combining ViSP with map-building incremental localization, also referred to as simultaneous localization and mapping (SLAM). Experimental results in both indoor and outdoor environments show sub-metre positional accuracies were achieved, while reducing the number of tracking losses throughout the image sequence. It is shown that by integrating model-based tracking with SLAM, not only does SLAM improve model tracking performance, but the model-based tracker alleviates the computational expense of SLAM's loop closing procedure to improve runtime performance. Experiments also revealed that ViSP was unable to handle occlusions when a complete 3D building model was used, resulting in large errors in its pose estimates. The second contribution of this dissertation is a novel map-based incremental localization algorithm that improves tracking performance, and increases pose estimation accuracies from ViSP. The novelty of this algorithm is the implementation of an efficient matching process that identifies corresponding linear features from the UAV's RGB image data and a large, complex, and untextured 3D model. The proposed model-based tracker improved positional accuracies from 10 m (obtained with ViSP) to 46 cm in outdoor environments, and improved from an unattainable result using VISP to 2 cm positional accuracies in large indoor environments.

The main disadvantage of any incremental algorithm is that it requires the camera pose of the first frame. Initialization is often a manual process. The third contribution of this dissertation is a map-based absolute localization algorithm that automatically estimates the camera pose when no prior pose information is available. The method benefits from vertical line matching to accomplish a registration procedure of the reference model views with a set of initial input images via geometric hashing. Results demonstrate that sub-metre positional accuracies were achieved and a proposed enhancement of conventional geometric hashing produced more correct matches - 75% of the correct matches were identified, compared to 11%. Further the number of incorrect matches was reduced by 80%.

ACKNOWLEDGMENTS

First and foremost, I am greatly thankful for my supervisor Dr. Costas Armenakis, who encouraged me to peruse graduate studies and has supported me throughout this degree. His motivation and guidance has made this experience very rewarding. I am indebted for all he has done for me.

I would like to thank my dissertation examination committee members, Dr. Sunil Bisnath, Dr. Gunho Sohn, Dr. Regina Lee, Dr. Michael Jenkin, and Dr. Ayman Habib for spending their time and energy to assess my work.

I thank the Natural Sciences and Engineering Research Council (NSERC) of Canada and the Ontario Graduate Scholarship (OGS) for financially supporting this research.

Special thanks are given to Professor James Elder's Human and Computer Vision Lab in York University's Centre for Vision Research for providing the UAV video data, and to the members of York University's GeoICT Lab who contributed to the generation of the York University 3D Virtual Campus Model: Damir Gumerov, Yoonseok Jwa, Brian Kim, and Yu Gao. I also thank Planning and Renovations, Campus Services & Business Operations at York University for providing the 3D model of the Bergeron Centre of Excellence in Engineering.

Last but not least, I would like to thank God and my family for their love and support: my parents Alain and Natalina, my brother Adrian, my sister Samantha, and my wife Annie and her family.

TABLE OF CONTENTS

ABSTRACT .... **ii**

ACKNOWLEDGMENTS .... **iv**

TABLE OF CONTENTS .... **v**

LIST OF TABLES .... **ix**

LIST OF FIGURES .... **xi**

LIST OF ACRONYMS .... **xviii**

CONVENTIONS .... **xxii**

DEFINITION OF TERMS .... **xxiii**

1 INTRODUCTION **1**

1.1 Current state of UAV navigation 1
1.2 Problem statement 3
1.3 Research motivation 5
1.4 Research contributions 7
1.5 Dissertation outline 9

2 COORDINATE SYSTEMS AND GEOMETRIC MODELS **11**

2.1 Object coordinate system 11
2.2 Body coordinate system 13
2.3 Camera coordinate system 16
2.4 Camera geometry 17
2.4.1 Interior orientation parameters 19
2.4.2 Exterior orientation parameters 21

3 FUNDAMENTALS OF UAV NAVIGATION **22**

3.1 Kinematic modelling 22
3.2 Inertial measurement unit (IMU) 24
3.3 Sensor augmentation 25

3.3.1 Position augmentation 25
3.3.2 Attitude augmentation 29
3.3.3 Optimal state estimation 32
3.4 Alternative navigation solutions for GNSS-denied environments 34
3.4.1 Interoceptive and exteroceptive measurement sensors 34

4 CAMERA-BASED LOCALIZATION 37

4.1 Map-based localization 37
4.1.1 Map-based incremental localization 39
4.1.2 Map-based absolute localization 43
4.1.3 Localization derived from target tracking 46
4.2 Map-building localization 48
4.3 Mapless localization 48
4.3.1 Optical flow 48
4.3.2 Appearance-based 49
4.3.3 Object recognition 49
4.4 Outdoor localization in structured environments 50
4.5 Unstructured outdoor localization 51

5 COMBINING MAP-BASED AND MAP-BULDING INCREMENTAL LOCALIZATION 52

5.1 Introduction to model-based tracking 54
5.1.1 Initialization and recovery 54
5.1.2 The tracking phase 55
5.1.3 Visual Servoing Package (ViSP) 56
5.1.4 TIN to polygon 3D model 61
5.2 Incremental map-building localization using structure from motion 63
5.2.1 Incremental Structure from Motion (iSfM) 64
5.2.2 Clustering Views for Multi-View Stereo (CMVS) 66
5.2.3 Patch-Based Multi-View Stereo (PMVS) 67
5.2.4 Photogrammetric Surface Reconstruction 67
5.2.5 Iterative Closest Compatible Point (ICCP) 68
5.3 Incremental map-building localization using RGB-D SLAM 69
5.4 Combining incremental map-based and map-building localization 71

5.5 Results: A UAV application of state-of-the-art map-based incremental localization .. 73

5.5.1 The UAV 73

5.5.2 The Pixhawk autopilot 74

5.5.3 Data collection and processing 74

5.6 Results: Combining ViSP and RGB-D SLAM 81

5.6.1 Experiment 1: Improving tracking maintenance 83

5.6.2 Experiment 2: Tracking pose across large areas 85

5.7 Concluding remarks 87

6 A MAP-BASED INCREMENTAL LOCALIZATION ALGORITHM 90

6.1 Pose Estimation Workflow 91

6.2 Feature selection 95

6.3 Extracting lines 97

6.4 Narrowing the search space in feature matching 100

6.5 Pose estimation via space resection 101

6.5.1 Point-based image orientation 102

6.5.2 Line-based image orientation 103

6.6 Results: A novel map-based incremental localization algorithm 108

6.6.1 Experiment 1: Map-based incremental localization in outdoor environments 109

6.6.2 Experiment 2: Map-based incremental localization in indoor environments 115

6.7 Concluding remarks 120

7 A MAP-BASED ABSOLUTE LOCALIZATION ALGORITHM 122

7.1 Introduction to geometric hashing 123

7.1.1 The pre-processing phase 125

7.1.2 The recognition phase 126

7.2 A map-based absolute localization solution for outdoor environments 129

7.2.1 The pre-processing phase 130

7.2.2 The recognition phase: Image-to-model registration 134

7.3 A map-based absolute localization solution for indoor environments 136

7.3.1 Feature selection 137

7.3.2 Synthetic panoramic images of the 3D model 138

7.3.3 Improving geometric hashing 141

7.4 Results: A novel map-based absolute localization algorithm 144
7.4.1 Experiment 1: A map-based absolute localization solution for outdoor environments 144
7.4.2 Experiment 2: A map-based absolute localization solution for indoor environments 151
7.5 Concluding remarks 153
8 CONCLUSIONS 156
9 FUTURE WORK 159
9.1 Cloud-based structure from motion 159
9.1.1 The RobotEarth Cloud Engine (RCE) 160
9.1.2 Mapping and tracking in the cloud 160
REFERENCES 162
APPENDIX A 181
A.1 Quaternion representation of attitude 181

LIST OF TABLES

Table 1.1. Common autopilots and fight controllers. 3

Table 1.2. Comparing the DJI Phantom 4 and the Inspire 1 (DJI, 2017). 5

Table 2.1. WGS84 ellipsoid parameters (NIMA, 2000).. 12

Table 3.1. GPS operation modes (10 km baseline).. 26

Table 3.2. Overview of indoor navigation technologies. Coverage refers to ranges of single nodes (Mautz, 2012). 36

Table 5.1. Mean and standard deviation for the difference in the UAV's positional coordinates from its onboard GPS and ViSP. 79

Table 6.1. Threshold values used in the outdoor and indoor experiments. 109

Table 6.2. Outdoor experiment: Average (σ) and standard deviation ($\sigma\sigma$) of the pose standard deviations from the least squares adjustment for the 5 sample frames in Figure 6.13. 111

Table 6.3. Outdoor experiment's check point statistics: The averages and standard deviations of the differences between ground truth model coordinates and object coordinate calculated by the point-to-line resection for horizontal lines and vertical lines. 111

Table 6.4. Accuracy assessment of the generated point clouds.. 114

Table 6.5. Indoor experiment: Average (σ) and standard deviation ($\sigma\sigma$) of the pose standard deviations from the least squares adjustment for the 5 sample frames in Figure 6.20. 117

Table 6.6. Indoor experiment's check point statistics: The averages and standard deviations of the differences between ground truth model coordinates and object coordinate calculated by the point-to-line resection for horizontal lines and vertical lines. 118

Table 6.7. Statistics describing the quality of the generated point cloud. The total number of points was 809861. The second row shows the percentage of points less than the maximum C2M distance specified in the first row. The third and fourth rows show the mean and standard deviation, respectively, of the C2M distances for the points referred to in the second row....... 120

Table 7.1. Table of NCC coefficients used to identify corresponding Vertical Line Features from the input panoramic image and synthetic panoramic image captured in Room B1.................... 143

Table 7.2: Map-based absolute localization threshold values used in the outdoor experiment.. 145

Table 7.3. Dataset 1: Camera pose standard deviations from the bundle adjustment, with the number of Ground Control Points (GCPs) and number of Tie Points (TP) per image............... 149

Table 7.4. Dataset 2: Camera pose standard deviations from the bundle adjustment, with the number of Ground Control Points (GCPs) and number of Tie Points (TP) per image............... 149

Table 7.5. Dataset 3: Camera pose standard deviations from the bundle adjustment, with the number of Ground Control Points (GCPs) and number of Tie Points (TP) per image............... 149

Table 7.6. Bundle adjustment statistics for object coordinates - Dataset 1: Number of images a tie point appeared in, number of tie points involved in the calculation, average coordinate standard deviations from the bundle adjustment, average coordinate differences compared to ground truth, and coordinate root mean square error when compared to ground truth. 150

Table 7.7. Bundle adjustment statistics for object coordinates - Dataset 2: Number of images a tie point appeared in, number of tie points involved in the calculation, average coordinate standard deviations from the bundle adjustment, average coordinate differences compared to ground truth, and coordinate root mean square error when compared to ground truth. 150

Table 7.8. Bundle adjustment statistics for object coordinates - Dataset 3: Number of images a tie point appeared in, number of tie points involved in the calculation, average coordinate standard deviations from the bundle adjustment, average coordinate differences compared to ground truth, and coordinate root mean square error when compared to ground truth. 150

Table 7.9. Map-based absolute localization threshold values used in the indoor experiment... 151

Table 7.10. Conventional geometric hashing: Truth table of the matching process................. 153

Table 7.11. Truth table of the matching process.. 153

LIST OF FIGURES

Figure 1.1. Applications of commercial small UAS (FAA, 2017). 1

Figure 1.2. UAVs registered by the FAA in 2016 (Skylogic Research, 2016)............................. 2

Figure 1.3. Overview of user requirements in terms of accuracy and coverage (Mautz, 2012).... 6

Figure 2.1. ECEF and local level coordinate systems. 11

Figure 2.2. IMU body frame. 14

Figure 2.3. Axial orientation in the body frame.. 14

Figure 2.4. The central perspective projection... 17

Figure 2.5: Types of radial lens distortion: Barrel distortion (left), pincushion distortion (right) (Ellum, 2001). 19

Figure 2.6. Decentring distortion (Ellum, 2001)... 20

Figure 3.1. Sample data observing static attitude (green is unfiltered, black is filtered)............ 31

Figure 3.2. Loosely-coupled sensor integration.. 33

Figure 3.3. Tightly-coupled sensor integration.. 33

Figure 3.4. Overview of indoor technologies in dependence on accuracy and coverage. 35

Figure 4.1. Types of camera-based localization solutions (DeSouza and Kak, 2002). 37

Figure 4.2. Example of edge-based tracking using a toy truck (Wuest and Stricker, 2007). 41

Figure 4.3. Generation of synthetic views on a view sphere centred on the 3D model at regularly sampled viewpoints (Petit, 2014)... 44

Figure 4.4. Three examples of coded targets used for point identification and camera calibration (Mautz, 2012)... 47

Figure 5.1. Combining map-based and map-building localization techniques.......................... 53

Figure 5.2. Sample frames demonstrating ViSP's model-based tracker (ViSP, 2013). 57

Figure 5.3. ViSP's pose estimation workflow. 58

Figure 5.4. Example of ViSP's initialization process. Left) 4 pre-specified 3D points on the wireframe model. Right) The user selects the 4 corresponding points on the image. ViSP uses these corresponding points to estimate the first camera pose. 58

Figure 5.5. Determining point position in the next image using the oriented gradient algorithm: A) calculating the normal at sample points, B) Sampling along the normal, C)-D) 2 out of the 180 3x3 predetermined masks, C) 180^o, D) 45^o (Modified from Comport et al., 2003). 59

Figure 5.6. An example of region growing and outline detection: The blue triangles belong to a group because one triangle is connected to at least one other triangle with a parallel normal. The outline of the group (black line) consists of the edges that belong only to one triangle. 62

Figure 5.7. Converting TIN building models to polygon models. A) The original TIN model of the York University's Lassonde Building. B) The resulting 3D polygon model of the Lassonde Building.. 62

Figure 5.8. Dense point cloud generation from UAV imagery.. 63

Figure 5.9. SfM software workflow (left) and dense point cloud generation (right).................. 64

Figure 5.10. Segmentation of the Lassonde Building model (left) and corresponding point cloud generated from the UAV data (right). Points are segmented into three groups based on their normal vector: Group 1 (red triangles): Normals pointing in the positive Y direction (±45^o), Group 2 (green triangles): Normals pointing in the negative X direction (±45^o), and Group 3 (blue triangles): normal pointing in the positive X direction (±45^o). 68

Figure 5.11. RGB-D SLAM's workflow for pose estimation and map creation. 70

Figure 5.12. Screen capture of RGB-D SLAM. The top window shows the map (RGB point cloud) being created. The bottom right and left windows show the SIFT image feature (red dots) and their scales (green circles), respectively.. 70

Figure 5.13. Model-based tracker / SLAM integration workflow. 72

Figure 5.14. Components of the developed indoor/outdoor UAV.. 75

Figure 5.15: Sensor integration strategy. 75

Figure 5.16. The UAV's flight path over the Lassonde Building... 76

Figure 5.17. York University's 3D campus model.. 77

Figure 5.18. Sample frames demonstrating ViSP’s model-based tracker. The West side (A-C) and North side (D-F) of York University’s Lassonde Building are being observed, the 3D building model is projected onto the image plane (red lines) using the respective camera intrinsic and extrinsic parameters. Harris corners (red crosses with IDs) are also being tracked. 78

Figure 5.19. Additional sample frames demonstrating ViSP’s model-based tracker. The East side of York University’s Lassonde Building is being observed, the 3D building model is projected onto the image plane (red lines) using the respective camera intrinsic and extrinsic parameters. Harris corners (red crosses with IDs) are also being tracked. 78

Figure 5.20. An overhead view of the UAV’s trajectory according to the onboard GPS (green) and ViSP solution (red)... 79

Figure 5.21. Positional comparison between the GPS solution and ViSP’s solution................. 80

Figure 5.22. Yaw angle comparison between the GPS solution and ViSP’s solution................ 80

Figure 5.23. ViSP / RGB-D SLAM integration workflow. 82

Figure 5.24. ViSP / RGB-D SLAM tree of coordinate transforms output from ROS’s *tf/view_frames*. 82

Figure 5.25. Initializing ViSP in the first experiment using the four corners of the door to the BEST lab. The 3D wireframe model of the door is projected onto the image (green lines) using the estimated camera pose. 83

Figure 5.26. Sample frames from the first experiment demonstrating ViSP’s model-based tracker aided by RGB-D SLAM. The door of York University’s BEST lab leaves the field of view in B) and E). The door re-enters the field of view in Frames C) and F), showing that tracking the pose was not lost. 84

Figure 5.27. A) RGB-D SLAM point cloud aligned with the 3D model using the pose estimates from ViSP. B) 3D model of a door (shown in red) used to generate the simplified rectangular model used in the first experiment. C) CloudCompare’s cloud-to-cloud (C2C) difference measurement between the 3D model of the door and the RGB-D SLAM point cloud. D) Histogram of C2C differences, with mean and standard deviation. 85

Figure 5.28. Left) The 3D indoor model of the second floor in the Bergeron Centre. The area of interest of the second experiment is outlined by the red box. Right) The 3D wireframe model used in the second experiment. 86

Figure 5.29. Sample frames from the second experiment demonstrating ViSP's model-based tracker aided by RGB-D SLAM. B) shows a misalignment between the image and the projected 3D model. This is due to error in the pose generated by RGB-D SLAM. However, RGB-D SLAM often corrected this error by closing small loops in every 10 frames. C) and E) show the portion of the 3D model that is behind the camera is inverted and projected onto the image plane. Tracking was not lost as ViSP is designed to recognize these points as outliers. 87

Figure 6.1. Simulated quadcopter in Gazebo's virtual environment (Open Source Robotics Foundation, 2017). 91

Figure 6.2. The trajectory of the UAV within the 3D CAD model of the building. 91

Figure 6.3. The proposed incremental map-based localization algorithm. 92

Figure 6.4. Line matching example: Real image (A) and corresponding synthetic image (B), each with 2 randomly sampled horizontal lines, sorted top to bottom (labelled 1 and 2) and 2 randomly sampled vertical lines, sorted left to right (labelled 3 and 4). C) shows the real image lines' corresponding virtual image line start and end points estimated by the line-to-point space resection. 95

Figure 6.5. Real image (left) and corresponding synthetic image (right). 96

Figure 6.6. (A) Top Left Corner templates, (B) Left Vertical Line Feature, ... 98

Figure 6.7. The extracted corners and Vertical Line Features from the edge image of the video frame. 99

Figure 6.8. Left (red) and Right (green) Vertical Line Features extracted from the synthetic image. 99

Figure 6.9. Horizontal shift between the Vertical Line Features of the video frame (red) and synthetic image (green). 100

Figure 6.10. Corresponding Vertical Line Features between the video frame (red) and synthetic image (green) are circled. 101

Figure 6.11. The four-parameter representation of a line in object space: Two positional parameters define the line's intersection with a plane perpendicular to it (X_s, Y_s) and two orientation parameters define the direction of line (α, θ) (Meierhold et al., 2008). 104

Figure 6.12. Point-to-line collinearity approach to establish relationships between object space lines and their partial projections in images (Meierhold et al., 2008). 106

Figure 6.13. Outdoor experiment: Tracking a 3D building model using the camera of a UAV. 111

Figure 6.14. VisualSFM and SURE used to generate a dense point cloud of the Lassonde Building from the UAV's video imagery. 113

Figure 6.15. Point cloud registered to the 3D using the pose estimated by the proposed MBT.113

Figure 6.16. Check points (red circles) for accuracy assessment of the point cloud. 113

Figure 6.17. Detecting the estimated points belonging (red) and not belonging (white) to the model... 114

Figure 6.18. Registration quality between the UAV's point cloud and the model. 115

Figure 6.19. The trajectory of the UAV (red circles) in the indoor environment without model-based tracking (left) and with model-based tracking (right). 116

Figure 6.20. Indoor experiment: Tracking a 3D indoor model using the camera of a UAV. 117

Figure 6.21. Top left) 3D Indoor model of the second floor of the study building. Top right) Point cloud captured using the integrated ViSP / RGB-D SLAM system. Bottom left) The collected point cloud overlaid on top of the 3D model. Bottom right) The cloud-to-mesh (C2M) distances (with 20 cm) between the collected point cloud and the 3D model. 119

Figure 7.1. Flow chart for the geometric hashing algorithm, modified from (Lamdan and Wolfson, 1988). 125

Figure 7.2. A) Model M1 with extracted Vertical Line Features (green lines). B) Incoming image with extracted Vertical Line Features (red lines). Establishing a canonical reference frame using C) a two-point basis (**P1** and **P2**) and the similarity transformation, D) a three-point basis (**P1**, **P2** and **P3**) and the affine transformation, and E) a four-point basis (**P1**, **P2**, **P3**, and **P4**) and the projective transformation. 128

Figure 7.3. A) The Lassonde Building model. B) The Lassonde Building model's normal vectors classified into four prominent groups. 131

Figure 7.4. A) Setting the virtual camera's position and orientation. B) Four synthetic images for the Lassonde Building model. 132

Figure 7.5. A) Model M1 Vertical Line Features 1 to 27, B) Model M4 Vertical Line Features 1 to 36. 133

Figure 7.6. Hash tables for A) Model M4 with Vertical Line Features 1 and 14 as the basis, i.e., (M4, (1, 14)), and B) Model M4 with Vertical Line Features 14 and 28 as the basis, (M4, (14, 28)). 134

Figure 7.7. A) Two images stitched together, B) the Canny edge image and the extracted Vertical Line Features, C) the hash table (M4, (14, 28)), which received the most votes, only the corresponding features are plotted. 136

Figure 7.8. Equally spaced grid of simulated cameras capturing 360° synthetic panoramic images in the Bergeron Centre 3D CAD model. 139

Figure 7.9. Generating a synthetic panoramic image: A) Synthetic images captured by the virtual camera at one grid location. B) Cylindrical projection of synthetic images (the radius of the cylinder is equal to the focal length of the camera). C) Resulting synthetic panoramic image. 139

Figure 7.10. A) Synthetic panoramic image of Room B1 with extracted Vertical Line Features. B) Incoming panoramic image of Room B1 with extracted Vertical Line Features. Establishing a canonical reference frame using C) similarity transformation: 2D hash table populated using as 2-point bases Vertical Line S2 from the synthetic panoramic image and Vertical Line R1 from the incoming panoramic image, D) affine transformation: 2D hash table populated using as 3-point bases Vertical Line S2 (and the top corner from its duplicate Line S11) from the synthetic panoramic image and Vertical Line R1 (and the top point from its duplicate Line R10) from the incoming panoramic image, and E) projective transformation: 2D hash table populated using as 4-point bases Vertical Line S2 (and its duplicate Line S11) from the synthetic panoramic image and Vertical Line R1 (and its duplicate Line R10) from the incoming panoramic image. F) Series generated from the Vertical Line Features of the model and G) the incoming image. Each element of the series is the number of Vertical Lines feature in the 1D hash table's bin. 140

Figure 7.11. Dataset 1: A) 6 images stitched together, B) the Canny edge image and the extracted Vertical Line Features, C) the hash table (M1, (1, 27)), which received the most votes, only the corresponding features are plotted. 146

Figure 7.12. Dataset 2: A) Four images stitched together, B) the Canny edge image and the extracted Vertical Line Features, C) the hash table (M4, (1, 14)), which received the most votes, only the corresponding features are plotted. 146

Figure 7.13. Dataset 3: A) 2 images stitched together, B) the Canny edge image and the extracted Vertical Line Features, C) the hash table (M4, (14, 28)), which received the most votes, only the corresponding features are plotted. 147

Figure 7.14. Camera positions for Datasets 1, 2, and 3. 148

Figure 7.15. Area of interest in the Bergeron Centre for the experiments. Panoramic images were captured in rooms B1, B2, B3, B4, B5, and B6. 152

Figure 7.16. Input panoramic images and corresponding synthetic panoramic images captured in the Bergeron Centre in A) Room B2, B) Room B3, C) Room B4, D) Room B5, and E) B6. Vertical Line Features extracted from synthetic panoramic images and input panoramic image are shown with green lines and red lines, respectively. 155

Figure 9.1. Blue rectangles depict ROS nodes, green rectangles for the RCE's endpoints, red rectangles for topics, small yellow squares for topic publishers, small blue squares for topic subscribers, small green squares for service servers, and small red squares for server clients. The direction of the arrows indicates the flow of data from the publishers to the topics and topics to subscribers, and arrows with labels connect service servers to service clients. Notably, this architecture was derived from the developers of the RCE (Gherardi et al., 2014). 161

LIST OF ACRONYMS

Acronyms are listed in alphabetical order.

2D - Two dimensional

3D – Three dimensional

AHRS - Attitude and Heading Reference System

AR – Augmented Reality

API - Application Program Interface

ASIFT – Affine Scale-Invariant Feature Transform

AVATAR - Autonomous Vehicle Aerial Tracking and Reconnaissance

BA – Bundle Adjustment

BEST - Bergeron Entrepreneurs in Science and Technology

BVLOS - Beyond Visual Line of Sight

CAD – Computer-Aided Design

CCD – Charge-Coupled Device

CPU - Central Processing Unit

CQAR - Closed Quarter Aerial Robot

CML - Concurrent Mapping and Localization

CORS - Continuously Operating Reference Station

CV – Computer Vision

DAvinCI - Distributed Agents with Collective Intelligence

DCM - Direction Cosine Matrix

DGPS - Differential Global Positioning System

DoF – Degrees of Freedom

DOT - Dominant Orientation Template

DSM - Digital Surface Model

ECEF - Earth-Centred-Earth-Fixed

EKF – Extended Kalman Filter

ENU - East-North-Up

EOP - Exterior Orientation Parameters

FAA - Federal Aviation Administration

FOV – Field of View

FPV – First Person View

FOG - Fibre Optic Gyroscope

g2o - General Graph Optimization

GCP – Ground Control Point

GNSS - Global Navigation Satellite System

GPS – Global Positioning System

GPU - Graphics Processing Unit

GSC - Geological Survey of Canada

HOG – Histograms of Oriented Gradient

ICP - Iterative Closest Point

ICCP - Iterative Closest Compatible Point

IGS - International GNSS Service

IMU – Inertial Measurement Unit

INS- Inertial Navigation System

IOP - Interior Orientation Parameters

IR - Infrared

iSfM - Incremental Structure from Motion

KLT - Kanade-Lucas-Tomasi

L2L - Line-to-Line

LAMBDA - Least-squares Ambiguity Decorrelation Adjustment

LDAHash - Linear Discriminant Analysis Hash

LEWIS - Library Entry Working through an Indexing Sequence

LIDAR - Light Detection and Ranging

LL – Local Level

LoD – Level-of-Detail

MEMS - Micro Electro-Mechanical Systems

MLAMBDA – Modified LAMBDA

MP – Megapixel

MVS - Multi-view Stereo

NCC – Normalized Cross-Correlation
NED - North-East-Down
P2L – Point-to-Line
P2P – Point-to-Point
PaaS - Platform-as-a-Service
PID – Proportional–Integral–Derivative (control loop feedback mechanism)
PMVS - Patch-Based Multi-View Stereo
POSIT - Pose from Orthography and Scaling with Iterations
PPP - Precise Point Positioning
PSD - Position Sensitive Device
PTAM - Parallel Tracking and Mapping
PVA – Position-velocity-acceleration
RADAR - Radio Detection and Ranging
RANSAC - Random Sample Consensus
RAPID - Real-time Attitude and Position Determination
R/C – Remote Control
RCE - RoboEarth Cloud Engine
RFID - Radio-Frequency Identification
RGB – Red, Green, Blue
RGB-D – Red, Green, Blue, Depth
RLG - Ring Laser Gyroscope
RMSE - Root Mean Square Error
ROS - Robot Operating System
RTH – Return to Home
RTK – Real-Time Kinematic
SaaS - Software-as-a-Service
SCERPO - Spatial Correspondence, Evidential Reasoning and Perceptual Organization
SfM – Structure from Motion
SGM - Semi-Global Matching
SIFT - Scale-Invariant Feature Transform
SLAM - Simultaneous Localization and Mapping

SSD- Sum of Squared Difference

SURE – Surface Reconstruction

SURF - Speeded Up Robust Features

TDOA - Time Difference of Arrival

TIN - Triangulated Irregular Network

TP – Tie Point

UAV – Unmanned Aerial Vehicle

US – United States

UTM - Universal Transverse Mercator

ViSP - Visual Servoing Platform

VPS - Vision Positioning System

VRML - Virtual Reality Modeling Language

VSRR - View Sequenced Route Representation

VVS - Virtual Visual Servoing

WAAS - Wide Area Augmentation System

WGS84 - World Geodetic System 1984

WLAN – Wireless Local Area Network

ZNCC - Zero-mean Normalized Cross Correlation

CONVENTIONS

Vectors: Vectors are shown using bold lowercase letters and symbols. Position vectors, indicated by '**r**', have a superscript and a subscript. The superscript indicates the coordinate frame of the vector. The subscript specifies the start and end point of the vector, separated by '/'. The starting point of the vector is not shown if it is the origin of the coordinate frame in which the vector is in. For example, $\mathbf{r}^{\mathbf{c}}_{\mathbf{a/b}}$ is the vector from point 'a' to 'b' in the 'c' frame. If point 'a' is the origin of the 'c' frame, then $\mathbf{r}^{\mathbf{c}}_{\mathbf{b}}$ signifies the position of 'b' in the 'c' frame.

Matrices: Matrices are shown using uppercase letters and symbols. Rotation matrices between coordinate systems, indicated by '**R**', have a superscript and a subscript. The subscript indicates the coordinate system before the rotation, and the superscript indicates the coordinate system after the rotation. For example, $\mathbf{R}^{\mathbf{b}}_{\mathbf{a}}$ is the rotation matrix that rotates vectors in the 'a' frame to the 'b' frame. The rotation matrices $\mathbf{R}_{\mathbf{x}}$, $\mathbf{R}_{\mathbf{y}}$, and $\mathbf{R}_{\mathbf{z}}$ indicate rotations about the x, y, and z axes, respectively.

DEFINITION OF TERMS

In his chapter prevalent technical terms for ‘positioning’ are defined and an explanation is given for technical terms frequently used in the field of navigation. In the literature, the process of determining a location is described by a number of different technical terms with slightly different meanings. The following definitions primarily reflect their usage in this work and might be defined slightly differently elsewhere.

Positioning

Positioning is the general term for determination of a position of an object or a person. It is particularly used to emphasize that the target object has been moved to a new location.

Tracking

The process of repeated positioning of a moving object or person over time is called tracking. Object tracking is also referred to as target tracking, path tracking, location tracking, mobile tracking, device tracking or asset tracking. Particular to this work is video tracking (also denoted as visual-, optical, photogrammetric, fiducial marker- or image feature tracking). Video tracking involves digital cameras and recognition of target objects in consecutive video frames.

Absolute and relative position

A distinction must be made between absolute and relative positions. Absolute position refers to a global or large area reference grid with its realization in the form of markers, landmarks or GNSS satellites. Absolute coordinate positions refer to a global reference system. In contrast, relative positions depend on a local frame of reference, e.g., coordinates within a small coverage area are given in delta positions to a local realization of reference.

Localization

Localization is the problem of estimating the pose (position and orientation) of the sensor relative to a map. Typically, one distinguishes between pose tracking, where the initial pose is known, global localization, in which no a priori knowledge about the starting position is given, and re-localization, where the pose estimate is incorrect or pose tracking is lost and recovered.

Geolocation

Geolocation is used for locating internet-connected devices where the determined location is descriptive or context based rather than a set of geographic coordinates.

Navigation

Navigation comprises 1) determination of position, speed, and attitude of a subject or object, 2) finding of the optimal path (in the sense of the fastest, shortest, or cheapest route) from a start to an end location, and 3) guidance along a given path and control of the difference between the current position and the planned path.

1 INTRODUCTION

Commercial unmanned aerial vehicles (UAVs) are presently used for numerous applications. Aerial photography and video applications account for 34% of usage, with construction, industrial, and real estate applications trailing as the second largest (FAA, 2017). Commercial UAVs typically carry cameras to capture aerial footage. That aerial footage typically includes weddings, sports, and outdoor family activities (Figure 1.1).

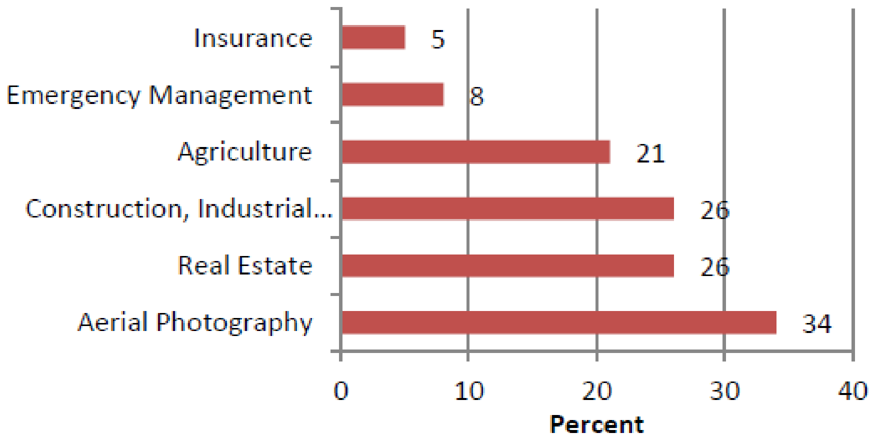


Figure 1.1. Applications of commercial small UAS (FAA, 2017).

1.1 Current state of UAV navigation

Figure 1.2 shows that as of April 2016, DJI has 50% of the UAV market share in North America of U.S. 3D Robotics comes in second place with a 7% share, trailed by Yuneec (Skylogic Research, 2016). The UAVs sold by the companies listed in Figure 1.2 are equipped with similar systems that are used to help the pilot keep the aircraft stable, and allow the aircraft to automatically navigate GPS waypoints. These systems are called flight controllers and autopilots, respectively. Table 1.1 provides a list of common flight controllers and autopilots.

An autopilot is a complete system that enables a UAV to navigate waypoints or return to home (RTH) for example. A flight controller is the system that keeps the UAV stable. Different systems have been developed for different types of flying. Namely, there are three different types of flying: 1) FPV (first person view) racing and freestyle, 2) aerial photography and videography, and 3) autonomous missions. Flight controllers and autopilots are designed to perform well at one specific flight style. The flight controller uses the sensor data to calculate the commands to send to the UAV in order for it to fly. Flight controllers are used for flight stabilization in the first two types of flying, namely FPV racing and aerial photography, where a pilot is always in control of the UAV. Autopilots are typically used in autonomous missions such as surveillance, inspection, and mapping.

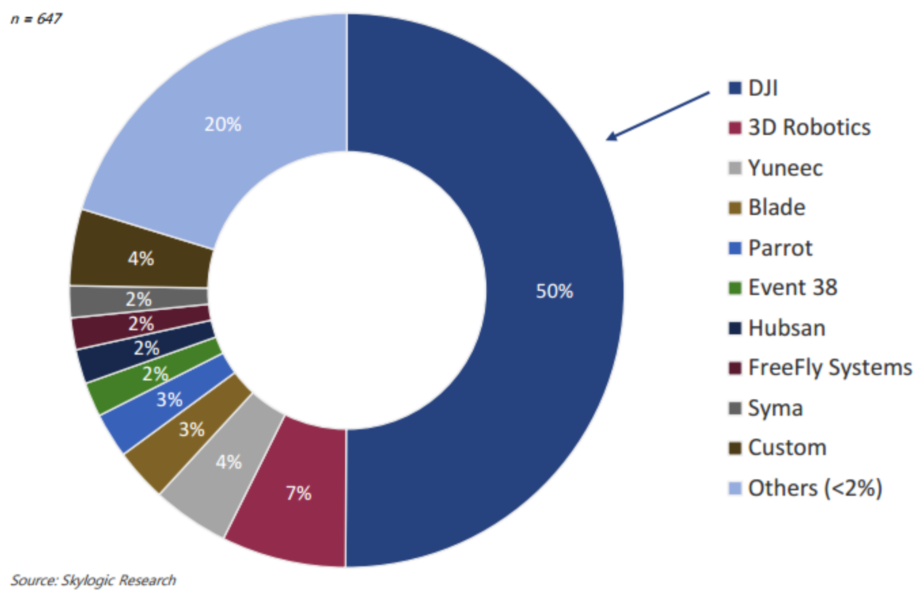


Figure 1.2. UAVs registered by the FAA in 2016 (Skylogic Research, 2016).

Flight controllers for racing or freestyle, for example the Flyduino KISS and the Lumenier LUX, are designed to recover from very fast roll rates, hold any angle, and be fully tuneable. The cost of these systems is low because they break from frequent collisions. Flight controllers designed for aerial photography and video, such as the DJI NAZA-M V2 and DJI A3, produce clear images and smooth video. The flight characteristics are smooth and the control stick rates are slow, resulting in slow manoeuvrability. Autonomous flying allows the pilot to fly a UAV without having to touch any of the controls. An autopilot used for autonomous flying has features like automatic take-off and landing, waypoint flying and data telemetry. Flight controllers and autopilots with open-source firmware allow users to add and improve features. DJI’s NAZA flight controllers have firmware that cannot be modified in any way by its users.

Alternatively, the 3DR Pixhawk from 3D Robotics is an autopilot designed specifically for Autonomous flying. The firmware is open-source, allowing the user to add features.

A typical autopilot consists of a processor, a power module, a data logging unit, R/C (remote control) inputs and outputs, telemetry and a ground station, a sensor suite, and software (PID controller, sensor drivers, navigation system, etc.). The sensor suite can consist of any combination of the following sensors: a GPS/GNSS module, an IMU (inertial measurement unit) consisting of a triad of accelerometers and gyroscopes, a compass (magnetometer), an optical camera, a barometer, an airspeed sensor, a distance sensor, and an optical flow sensor.

Table 1.1. Common autopilots and fight controllers.

Autopilot/Flight controller	**Flying type**	**Price**
Flyduino KISS (kiss.flyduino.net/)	Racing or freestyle	US$45
Lumenier LUX (lumenier.com/products/flight-controllers)	Racing or freestyle	US$40
DJI NAZA-M V2 (dji.com/naza-m-v2)	General use or photography	US$300
DJI A3 (dji.com/a3)	Professional photography	US$900
3DR PixHawk (pixhawk.org/modules/pixhawk)	DIY and autonomous	US$204

1.2 Problem statement

Figure 1.2 shows that the majority (34%) of commercial UAV usage in the United States is for photography and video applications, typically for weddings, sports, and outdoor activities. The need for precise and reliable navigation is increased in these populated urban environments, where the possibility of crashing is high, as UAVs fly at low altitudes among buildings, avoid obstacles, and perform sharp manoeuvres. This need further increases in GNSS-denied environments, and in dense GNSS signal multipath environments such as indoors and in urban canyons. In these environments, the positioning accuracy provided by GNSS degrades significantly because of cycle slips, signal blockage and multipath, and poor satellite geometry (Hofmann-Wellenhof et al., 2001).

Regarding attitude estimation, errors are propagated from gyroscope, accelerometer, and magnetometer measurements. The attitude solution contains systematic errors, such as biases and scaling errors, and magnetometer soft iron and hard iron errors, which are due to nearby magnetic fields from permanent magnets, electric currents, or large iron bodies. Reported attitude accuracies of low-cost MEMS sensors are 0.02^o to 5^o. Notably, the accuracy of these sensors is dependent on temperature, and the accuracy decreases as the tilt angle increases. However, this error is limited since the roll and pitch for stabilized UAVs are not likely to exceed $\pm 20^o$. The second limitation is that the magnetometer operates at a lower sampling interval than the IMU, so the azimuth updates are less frequent. Lastly, the attitude can be determined only if the platform's velocity is constant - the accelerometers observe gravity alone. More accurate and reliable attitude estimation may be required when navigating in dense urban environments in order to reliably avoid obstacles and navigate waypoints.

In the most recent developments to address these issues, industry leader DJI revealed its Vision Positioning System (VPS) in 2017. VPS is a technology that helps a quadcopter UAV hover while flying indoors without a GNSS signal. By using dual downward-facing cameras and dual ultrasonic distance sensors the system can scan the ground below (up to 10 metres) and helps the drone hold its position while inside a building. This technology is targeted for filming indoors, where the user controls the camera and while the drone hovers on its own. The specifications for DJI's top selling drones, the Phantom 4 and the Inspire 1 are provided in Table 1.2. Although these UAVs can precisely hover in one location in GNSS-denied environments, the effective range of the VPS is limited (2.5 metres for the Inspire 1 and 10 metres for the Phantom 4). However, there is still no capability to fly autonomously without GNSS.

Mautz's 2012 habilitation (post-doctoral) dissertation provides a detailed list of localization technologies designed for GNSS-limited and GNSS-denied environments, along with applications, and each with technical and user requirements. Further, a description of technologies and measuring principles are provided. Mautz's literature review concludes that current solutions cannot cope with the performance level that significant applications require. Apart from insufficiency in position accuracy, coverage and availability, the need for extensive infrastructure deployment and maintenance is the main reason why system implementations are not sufficiently economical. A good fraction of research approaches are also missing appealing usability to enable wide-scale consumer adoption. Microsoft's Indoor Localization Competition 2016 concluded with: "Over the last 15 years, several indoor localization technologies have been proposed and experimented by both academia and industry, but we have yet to see large scale deployments" (Microsoft, 2016).

1.3 Research motivation

According to Mautz (2012), film and photography is categorized under *mass market applications*. Other mass market applications include pedestrian navigation, robot navigation, augmented reality, and other location-based services. Figure 1.3 provides common applications along with their accuracy and coverage requirements.

Table 1.2. Comparing the DJI Phantom 4 and the Inspire 1 (DJI, 2017).

	DJI Inspire 1	**DJI Phantom 4**
Price	US$2900	US$1400
Number of rotors	4	4
Applications	Film and Photography	Film and Photography
Maximum flight time	18 minutes	28 minutes
Still camera resolution	12 megapixels	12 megapixels
Weight	2935 grams	1380 grams
Payload capacity	1700 grams	462 grams
Maximum speed	22 m/s	20 m/s
Maximum flying altitude	4500 metres	6000 metres
Battery capacity	4500 mAh	5350 mAh
Battery weight	570 grams	462 grams
Battery type	Lithium polymer 6S (22V)	Lithium polymer 4S (15.2V)
Vision positioning maximum altitude	5 metres	10 metres
Vision positioning maximum range	2.5 metres	10 metres
Vision positioning maximum velocity	8 m/s	10 m/s
Vertical hovering accuracy	+/-0.5 m	+/- 0.1 m (when Vision Positioning is active) or +/-0.5 m.
Horizontal hovering accuracy	+/-2.5 m	+/- 0.3 m (when Vision Positioning is active) or +/-1.5 m.

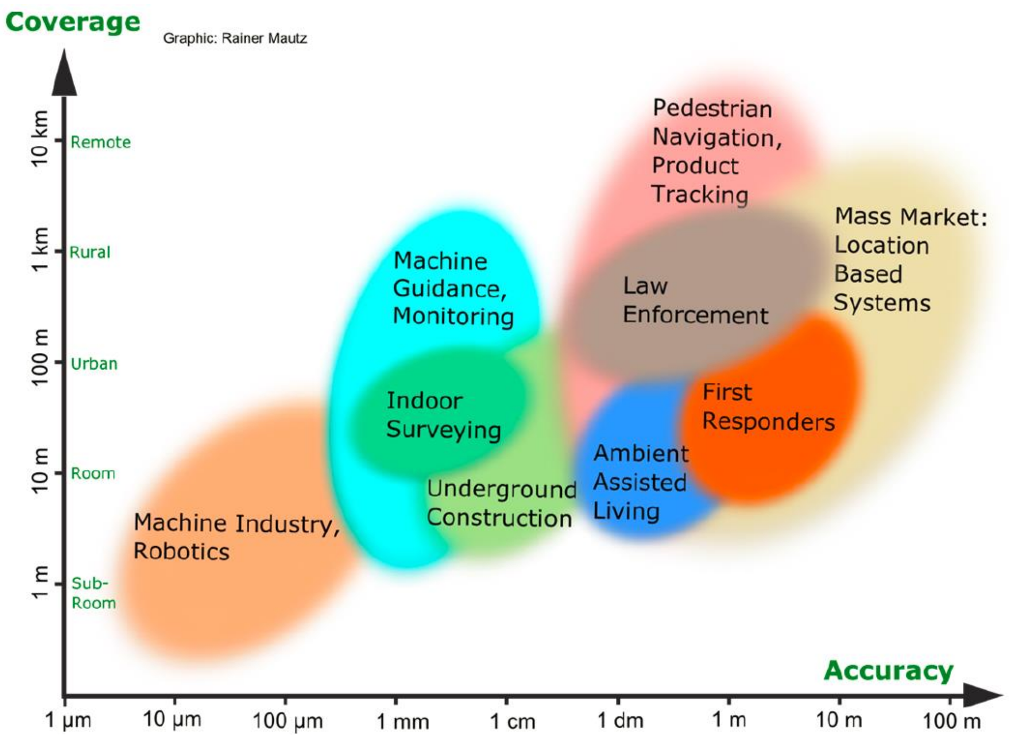


Figure 1.3. Overview of user requirements in terms of accuracy and coverage (Mautz, 2012).

The general user requirements for mass-market localization have been provided by Wirola et al. (2010): Mass market applications for navigation in GNSS-limited and GNSS-denied environments require the use of standard devices without supplementary physical components. Further requirements include:

- Horizontal accuracy of about 1 metre.
- Update rate of 1 Hz.
- No perceivable latency.
- Scalable without loss of accuracy.
- Functional within all types of buildings.
- Stability against environmental changes.
- No coverage gaps.
- Greater than 99 % availability.
- Moderate costs (set-up, per user device, per room, maintenance, and training).
- No or little infrastructure (e.g., markers, passive tags, active beacons).

Mautz (2012) advocates that solutions not requiring a fixed infrastructure should be preferred. The monetary cost should be proportional to the number of users, rather than to the coverage area. Further, the environment is constantly changing as new structures, modules and equipment, furniture are installed. These environmental changes introduce additional challenges to all fingerprinting techniques, i.e., those that are based on comparing the properties of magnetic field, radio signal propagation, soundscape or other signals to a predefined map of such signals.

The objectives of this dissertation are to 1) improve the localization accuracy for UAVs flying in GNSS-limited environments, and 2) provide a localization solution for UAVs flying in GNSS-denied environments. The localization solution will enable automatic way-point navigation in GNSS-limited and GNSS-denied environments and satisfy the mass market requirements specified by Wirola et al. (2010). The localization solution may also be used to provide pilots with the telemetry data required for beyond visual line of sight (BVLOS) UAV flights in GNSS-denied environments.

1.4 Research contributions

Among the solutions presented by Mautz (2012), the general consensus is that camera-based indoor localization solutions have the most potential to serve the mass market. Low-cost cameras achieve accuracy levels between tens of micrometres and decimetres. The coverage areas of the systems reviewed range from 4 m^2 to large room sizes, but can be scaled arbitrarily. High update rates of typically more than 10 Hz allow for kinematic applications such as precision-navigation, real-time mapping and pose estimation.

Chapter 4 explores the multitude of approaches to camera-based localization. Mapless localization, using optical flow to produce visual odometry for example, is subject to drift. But this drift is eliminated using map-building techniques like visual SLAM, as it distributes this error using loop closures, this task is however computationally expensive. Map-based methods are more efficient because they close loops using a pre-exiting map, thus the amount of data and the amount of time spent collecting data are reduced because there is no need to re-observe the same areas multiple times. However, despite being in development since the 1990's, map-based localization techniques have only been applied to using small and simple maps, in small and uncluttered environments (Lahdenoja et al., 2015). This dissertation first assesses of state-of-the-art map-based localization techniques in larger environments, using larger, more complex maps. Specifically, the state-of-the-art map-based incremental localization is applied in two scenarios:

1) UAV navigation in GNSS-denied outdoor environments using a georeferenced 3D building model.
2) UAV navigation in indoor environments using a georeferenced 3D indoor building model.

In both situations, the localization solution outputs the camera's pose for each image frame with respect to the 3D model's georeferenced coordinate system. The assessment of the results revealed the shortcomings of the state-of-the-art in map-based localization, which explained why the maps used, and the working environments, in previous literature were limited in size and complexity.

The first contribution of this work is **an incremental localization solution that combines map-based and map-building localization.** This solution leverages the complementary nature of the two localization techniques to obtain improved performance in terms of pose accuracy and tracking maintenance.

The second contribution of this work is **a map-based incremental localization algorithm** that overcomes the deficiencies of the state-of-the-art. The experiments demonstrate two novel applications:

1) A UAV localization technology for bridging GNSS gaps and aiding other navigation sensors (GNSS/INS) in outdoor environments.
2) A building-scale indoor localization solution for UAVs.

Lahdenoja et al. (2015) identified that the initialization and recovery phases of map-based incremental localization techniques is often done manually. However, when this process is automated via map-based absolute localization methods, also referred to as *tracking-by-detection*, either fiducials (e.g., landmarks or markers) or the object's texture is used. The third contribution of this work is **a map-based absolute localization algorithm** that is based purely on natural geometric features. Using geometric features (i.e., point, lines, polygons, etc.) is more robust solution than texture-based features (i.e., SIFT (Lowe, 2004), SURF (Bay et al., 2008), etc.) because shadows, changing illumination conditions, changing colours, etc., have less influence on localization performance. Further, by using natural features (e.g., lines present in the environment) there is no need to install and maintain infrastructure or targets throughout the environment. The experiments demonstrate the application of map-based absolute localization in both indoor and outdoor environments.

1.5 Dissertation outline

Chapter 2 details the geometric models used in this work. Additionally, the coordinate systems are defined. Chapter 3 explains the fundamentals of UAV navigation, specifically detail is provided about the autopilot technologies that are used on most common UAV platforms. The focus is on the sensors and software used to process the sensor data to provide a navigation solution. Chapter 4 provides an overview of camera-based localization solutions that may be used to increase the accuracy and performance of UAV localization in GNSS-limited and GNSS-denied environments.

Chapter 5 presents the first contribution of this dissertation. An incremental localization solution is proposed that combines map-based and map-building localization techniques. The chapter first introduces the state-of-the-art in map-based localization, then map-building localization. The chapter then demonstrates how these two localization techniques are complementary and the performance of the integrated solution is superior to each solution's individual performance.

Chapter 6 presents the second contribution of this dissertation. A map-based incremental localization algorithm is proposed. The results demonstrate an improvement over the state-of-the-art in terms of camera pose estimation accuracy in both outdoor and indoor environments.

Chapter 7 provides the third contribution of this dissertation. A map-based absolute localization algorithm is presented. This solution is used to automatically initialize map-based incremental localization methods. Initialization is often done manually, but when it has been automated, targets and texture-based natural features have been used. The proposed algorithm is based on matching natural geometric features, specifically vertical lines appearing in the environment. The results demonstrate the advantage of geometric features over targets and texture-based features in terms of robustness in changing environmental conditions (e.g., lighting and shadows), cost of preparation and maintenance. Finally, Chapter 8 provides a summary of conclusions and 9 proposes future work.

2 COORDINATE SYSTEMS AND GEOMETRIC MODELS

This chapter details the coordinate systems, geometric models, and conventions throughout this dissertation. Traditional photogrammetry commonly uses two coordinate systems, one in the camera frame and the other in object space or mapping frame. A third coordinate system is required for the navigation system, referred to as the body, or navigation, coordinate system.

2.1 Object coordinate system

The object space, or mapping coordinate system, is defined in the local-level (LL) frame, also known as the North-East-Down (NED) or local-tangent frame. This is a right-handed coordinate system with an origin at some position within the scene, defined in the Earth-Centred-Earth-Fixed (ECEF) geodetic coordinate system. The X and Y axes point along the North and East directions respectively, and form a plane tangent to the surface of the World Geodetic System 1984 (WGS84) Earth ellipsoid. The Z-axis is normal to the ellipsoid, completing the right-handed system, as illustrated in Figure 2.1.

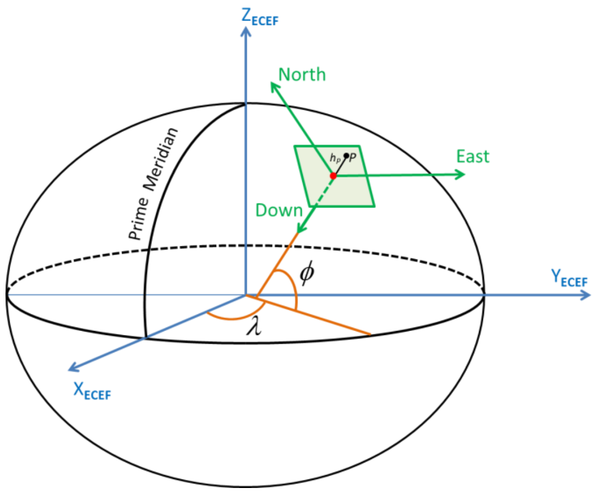


Figure 2.1. ECEF and local level coordinate systems.

Several geodetic transformations are commonly used in UAV navigation. Positioning provided by GNSS is usually given in latitude, longitude and height above the ellipsoid. To convert these values to the local-level frame, it is first necessary to express the GNSS position in ECEF coordinates using:

(2.1)

$$\begin{bmatrix} X_P \\ Y_P \\ Z_P \end{bmatrix}^{ECEF} = \begin{bmatrix} (N + h_P)\cos(\phi_P)\cos(\lambda_P) \\ (N + h_P)\cos(\phi_P)\sin(\lambda_P) \\ \left(\frac{Nb^2}{a^2} + h_P\right)\sin(\phi_P) \end{bmatrix}$$

where (ϕ_P, λ_P, h_P) are the geodetic coordinates (latitude, longitude, height) of point P. N is the called the Normal, which is the radius of curvature of the prime vertical, expressed as:

(2.2)

$$N = \frac{a}{\sqrt{1 - e^2 \sin^2(\phi_P)}}$$

The WGS84 ellipsoid parameters are given in Table 2.1.

Table 2.1. WGS84 ellipsoid parameters (NIMA, 2000)

Parameter	**Notation**	**Value**
Semi-major axis	a	6378137.000 m
Semi-minor axis	b	6356752.314245 m
Eccentricity	e	1/298.257223563

The ECEF coordinates are then converted to local-level north-east-down (NED) coordinates by rigid body transformation:

(2.3)

$$\begin{bmatrix} N_P \\ E_P \\ D_P \end{bmatrix}^{LL} = \mathbf{R}_{ECEF}^{LL} \begin{bmatrix} X_P - X_o \\ Y_P - Y_o \\ Z_P - Z_o \end{bmatrix}^{ECEF}$$

(2.4)

$$\mathbf{R}_{ECEF}^{LL} = \mathbf{R}_x\left(\phi_o - \frac{\pi}{2}\right)\mathbf{R}_z(\lambda_o) = \begin{bmatrix} -\sin(\phi_o)\cos(\lambda_o) & -\sin(\phi_o)\sin(\lambda_o) & \cos(\phi_o) \\ -\sin(\lambda_o) & \cos(\lambda_o) & 0 \\ -\cos(\phi_o)\cos(\lambda_o) & -\cos(\phi_o)\sin(\lambda_o) & -\sin(\phi_o) \end{bmatrix}$$

where $\mathbf{R}_{ECEF}^{LL}$ is the rotation matrix relating the ECEF frame to the local-level frame, and $(X_o, Y_o, Z_o)^{ECEF}$ is the origin of the local-level frame, defined in ECEF coordinates. Notably, in this dissertation, the X, Y, and Z axes are arranged to the form the right-handed NED coordinate system instead of the east-north-up (ENU) system. The NED convention is more prevalent in research, especially in aerial applications, where down is defined as positive because in an aerial survey most objects of interest are below the mapping sensor. Further, the NED axes coincide with the vehicle-fixed roll-pitch-yaw axes when the vehicle is level and pointing north.

A common alternative uses a conformal mapping projection such as Universal Transverse Mercator (UTM) for the object space coordinate system to reduce the computation cost of generating output maps (Legat, 2006). However, the curvature of the earth and the vertical relief of the terrain introduce errors into the solution (Ressl, 2002). To avoid this complication, the geometry is reconstructed in a local-level coordinate system defined at the centre of the scene, and subsequently transforming to a mapping projection using geodetic methods. Skaloud and Legat (2007), Legat (2006), and Ressl (2002) examine of the problems associated with using a conformal mapping projection. In general, these effects are minimal over small land areas when the terrain is essentially at sea level.

2.2 Body coordinate system

The body (b), or navigation, coordinate system is defined by the body frame composed of the sensing axes of the IMU. This system is idealized in the sense that the accelerometer and gyroscope triads are not perfectly aligned in space and are not truly orthogonal (Schwarz and Wei, 2000). The origin is defined at the convergence of these axes, and is illustrated in Figure 2.2. The origin is denoted in the object space by $\mathbf{r}_b^M = \begin{bmatrix} X_b & Y_b & Z_b \end{bmatrix}^T$.

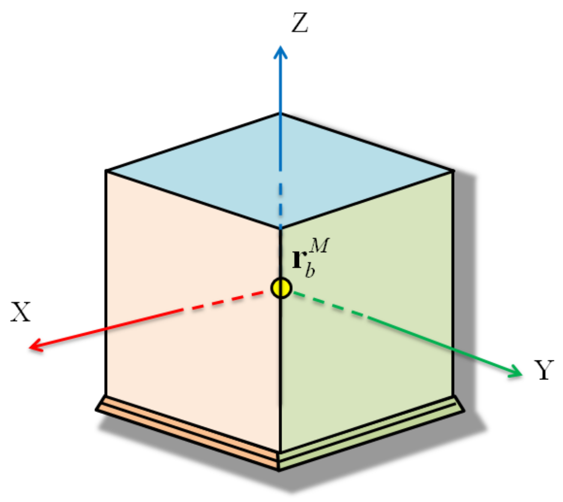


Figure 2.2. IMU body frame.

Two right-handed axis definitions are common in navigation; one with the Z-axis down and the X-axis to the front of the device (Titterton and Weston, 1997), and the other with the Z-axis up and the Y-axis directed forward (Swartz and Wei, 2000). In both cases, orientations about the longitudinal, transversal, and vertical axes are defined using roll (φ), pitch (θ), and yaw (ψ) angles, respectively. As it is more intuitive to have the roll about the X-axis, the former convention was chosen, as seen in Figure 2.3.

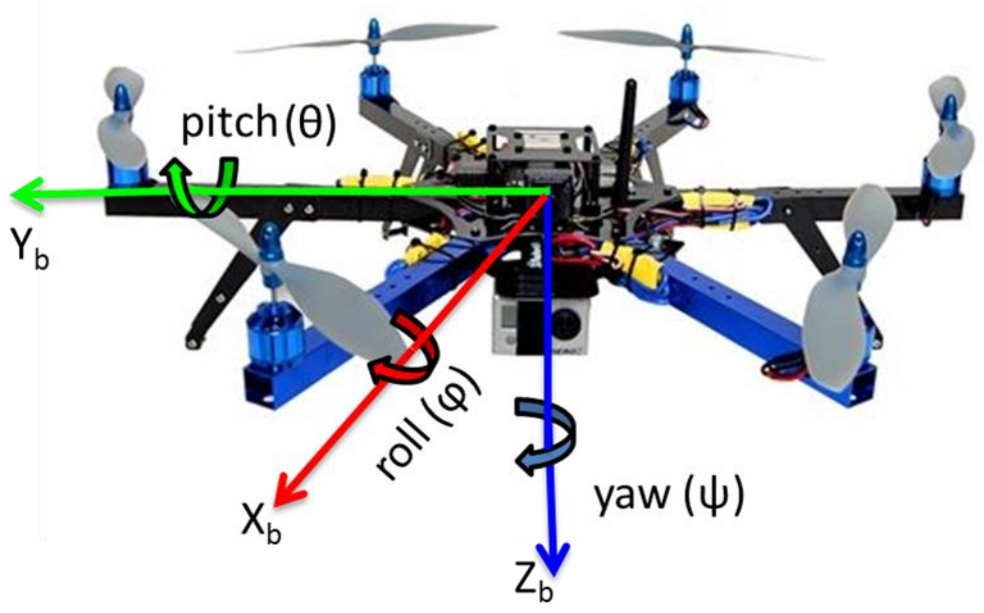


Figure 2.3. Axial orientation in the body frame.

The equivalent direction cosine matrix (DCM) is given by:

$$\mathbf{R}_M^b = \mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi) \tag{2.5}$$

where

$$\mathbf{R}_x(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & \sin(\varphi) \\ 0 & -\sin(\varphi) & \cos(\varphi) \end{bmatrix} \tag{2.6}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.7}$$

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

Equation (2.5) becomes:

$$\mathbf{R}_M^b = \begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\theta) \\ \sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi) & \sin(\phi)\sin(\theta)\sin(\psi) + \cos(\phi)\cos(\psi) & \sin(\phi)\cos(\theta) \\ \cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) & \cos(\phi)\cos(\theta) \end{bmatrix} \tag{2.9}$$

It should be noted that although the axes of the local-level frame used as the mapping frame are assumed to be exactly parallel and have the same direction as the axes of the local-level frame defined by a level IMU pointing north, this situation is never the case (unless the origins of the two exactly coincide) because different points on the ellipsoid will have different normal vectors and different directions to North. Fortunately, the error from this misalignment is negligible and can be ignored when the two origins are close. However, this error can be significant if the survey area is large, in this case the local-level mapping frame should not be used, alternatively multiple local-level frames could be used in order to keep the origins of the mapping and body frames within acceptable proximity.

2.3 Camera coordinate system

The camera coordinate system is right-handed and defined with an origin at the perspective centre of the camera. The X and Y axes define a plane parallel to the imaging sensor and correspond to the rightward and upward directions, respectively, when holding the camera as if taking a picture. The Z-axis of the image coordinate system completes the right-hand coordinate system and extends positively in the direction of the photographer. The intersection of the Z-axis of the imaging coordinate system (the 'optical axis') with the plane of the image sensor is the principal point. To avoid the complication of dealing with an inverted image, the image sensor is virtually located 'in front' of the perspective centre, as seen in Figure 2.4. The position vector of the perspective centre of the camera in the object space is defined as $\mathbf{r}_c^M = [X_c \quad Y_c \quad Z_c]^T$ and a position vector for any feature in the object space, point P, is $\mathbf{r}_P^M = [X_P \quad Y_P \quad Z_P]^T$, corresponding to a feature in the camera frame, $\mathbf{r}_p^c = [x_p \quad y_p \quad -f]^T$, where *f* is the calibrated focal length, which is the distance from the rear nodal point of the lens to the principal point,. Notably, image points are measured in the image (i) frame, where the origin is the top left corner of the image and the units are in pixels. The image point's X and Y coordinates represent the column (c) and row (r), respectively. Image measurements are transformed from the image coordinate system to the camera coordinate system using:

$$\begin{aligned} x_p &= (c_p - o_x)s_x \\ y_p &= -(r_p - o_y)s_y \end{aligned} \tag{2.10}$$

where (o_x, o_y) are the pixel coordinates of the principal point, (s_x, s_y) are the effective pixel horizontal and vertical dimensions.

The orientations of the image coordinate system with respect to the object coordinate system can be defined equivalently by Euler angles, DCM, or quaternions (Kuipers, 1999). The convention commonly used in photogrammetry is the omega-phi-kappa, where the rotations are expressed using Euler angles and computed using a DCM. Thus, the rotation matrix from the object coordinate system to the camera coordinate system is denoted by $\mathbf{R}_M^c$, and is given by:

$$\mathbf{R}_M^c = \mathbf{R}_z(\kappa)\mathbf{R}_y(\phi)\mathbf{R}_x(\omega) \tag{2.11}$$

Using the rotation matrices from Equation (2.6) to Equation (2.8), the rotation matrix becomes:

$$\mathbf{R}_M^c = \begin{bmatrix} \cos(\phi)\cos(\kappa) & \sin(\omega)\sin(\phi)\cos(\kappa)+\cos(\omega)\sin(\kappa) & -\cos(\omega)\sin(\phi)\cos(\kappa)+\sin(\omega)\sin(\kappa) \\ -\cos(\phi)\sin(\kappa) & -\sin(\omega)\sin(\phi)\sin(\kappa)+\cos(\omega)\cos(\kappa) & \cos(\omega)\sin(\phi)\sin(\kappa)+\sin(\omega)\cos(\kappa) \\ \sin(\phi) & -\sin(\omega)\cos(\phi) & \cos(\omega)\cos(\phi) \end{bmatrix} \tag{2.12}$$

2.4 Camera geometry

The camera geometry is described by a perspective projective transformation (Wolf and Dewitt, 2000). The model, shown in Figure 2.4, assumes the camera is an ideal pinhole camera with an infinitesimally small lens. A single ray of light from every visible point in the object space passes through the lens, onto the image plane.

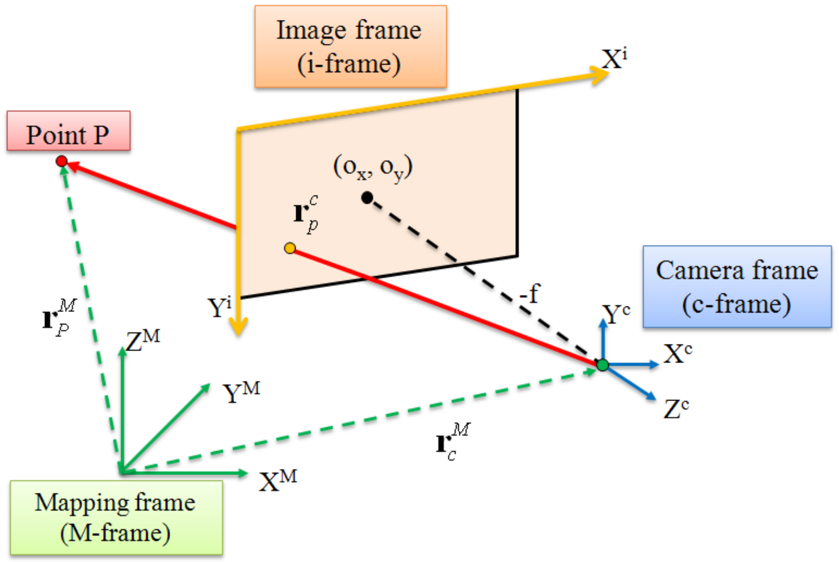


Figure 2.4. The central perspective projection.

In this model, a seven-parameter conformal transformation relates mapping coordinates $\left(X_P, Y_P, Z_P\right)^M$ with its camera coordinates $\left(x_p, y_p, -f\right)^c$, as given by:

(2.13)

$$\begin{bmatrix} x_p \\ y_p \\ -f \end{bmatrix}^c = \mu \mathbf{R}_M^c \begin{bmatrix} X_P - X_c \\ Y_P - Y_c \\ Z_P - Z_c \end{bmatrix}^M$$

where μ is the scale between the mapping frame and camera frame for point P, and $\mathbf{R}_M^c$ is the rotation matrix between the mapping frame and the camera frame as given in Equation (2.11). Rearranging the third equation in (2.13):

(2.14)

$$\mu = \frac{-f}{r_{31}\left(X_P - X_c\right) + r_{32}\left(Y_P - Y_c\right) + r_{33}\left(Z_P - Z_c\right)}$$

Substituting this into the first and second equations of Equation (2.13) yields:

(2.15)

$$\begin{aligned} x_p &= -f\,\frac{r_{11}(X_P - X_c) + r_{12}(Y_P - Y_c) + r_{13}(Z_P - Z_c)}{r_{31}(X_P - X_c) + r_{32}(Y_P - Y_c) + r_{33}(Z_P - Z_c)} \\ y_p &= -f\,\frac{r_{21}(X_P - X_c) + r_{22}(Y_P - Y_c) + r_{23}(Z_P - Z_c)}{r_{31}(X_P - X_c) + r_{32}(Y_P - Y_c) + r_{33}(Z_P - Z_c)} \end{aligned}$$

These equations assume that an object point, the corresponding image point, and the perspective centre of the camera are collinear, hence they are termed the collinearity equations. In reality, the pinhole lens is replaced by a compound lens system to reduce the exposure time of the film or CCD chip. Due to manufacturing imperfections, the lenses cause distortions and the image plane is neither perfectly flat nor perpendicular to the optical axis. Fortunately, these deviations can be modelled through geometric camera calibration. Accurately determining the values for these parameters is important because they affect the object coordinates of points derived from the images.

2.4.1 Interior orientation parameters

The camera's interior orientation parameters (IOPs) refer to the focal length, the principal point offsets (x_o, y_o), and the lens distortion parameters of the camera. The principal point offsets are the physical offsets between the geometric centre of the sensor and the optical axis of the lens (Clarke et al., 1998). They are used to translate image points from the image coordinate system to the camera coordinate system. Radial lens distortion is the result of imperfections in the grinding of the camera lenses. This distortion is common and is usually the largest source of error among the image coordinate dependent errors for non-metric cameras (Wackrow, 2008). The two types of radial lens distortion are commonly known as barrel and pincushion distortion, as seen in Figure 2.5.

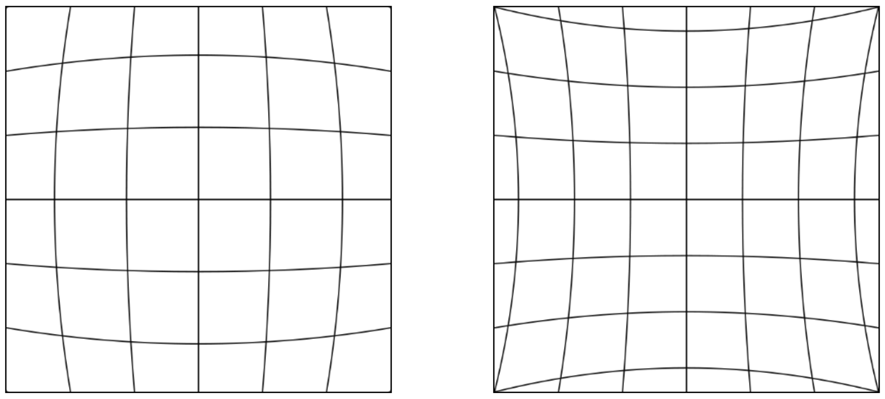


Figure 2.5: Types of radial lens distortion: Barrel distortion (left), pincushion distortion (right) (Ellum, 2001).

Typically, radial lens distortion δr is modelled with a Seidel (odd-ordered polynomial) series truncated at the second or third term (Fraser, 1997)

$$\begin{aligned}
\delta x_r &= \bar{x}\frac{\delta r}{r} \\
\delta y_r &= \bar{y}\frac{\delta r}{r} \\
\delta r &= k_1 r^3 + k_2 r^5 + k_3 r^7 + \ldots \\
r &= \sqrt{\bar{x}^2 + \bar{y}^2}
\end{aligned} \tag{2.16}$$

where k_i are the coefficients of radial distortion, $\bar{x}$ and $\bar{y}$ are the distances from the principal point, calculated as $\bar{x} = x - x_o$ and $\bar{y} = y - y_o$, respectively. Notably, the radial lens distortion parameters change as the focal length changes.

Decentreing, or tangential, distortion is caused by the misalignment of the axes of the individual lenses, and when the image plane is not perpendicular to the camera's optical axis (Brown, 1966). The distortion effects on the image as illustrated in Figure 2.6.

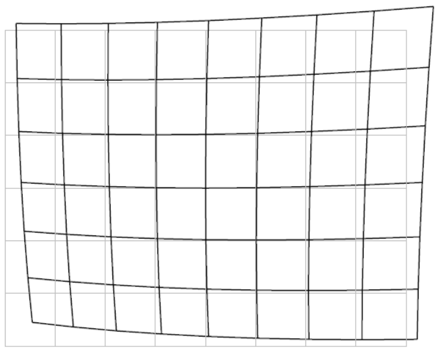


Figure 2.6. Decentring distortion (Ellum, 2001).

This distortion is often modelled as the Conrady-Brown model:

(2.17)

$$\begin{aligned} \delta x_d &= p_1\left(r^2 + 2\bar{x}^2\right) + 2p_2\bar{x}\bar{y} \\ \delta y_d &= p_2\left(r^2 + 2\bar{y}^2\right) + 2p_1\bar{x}\bar{y} \end{aligned}$$

where p_i are the decentring distortion parameters. The collinearity equations are extended to include the interior orientation and take the form:

(2.18)

$$\begin{aligned} x_p &= x_o + \delta x - f\,\frac{r_{11}(X_P - X_c) + r_{12}(Y_P - Y_c) + r_{13}(Z_P - Z_c)}{r_{31}(X_P - X_c) + r_{32}(Y_P - Y_c) + r_{33}(Z_P - Z_c)} \\ y_p &= y_o + \delta y - f\,\frac{r_{21}(X_P - X_c) + r_{22}(Y_P - Y_c) + r_{23}(Z_P - Z_c)}{r_{31}(X_P - X_c) + r_{32}(Y_P - Y_c) + r_{33}(Z_P - Z_c)} \end{aligned}$$

where the combined lens distortion is expressed as:

$$\begin{aligned}\delta x &= k_1\bar{x}r^2 + k_2\bar{x}r^4 + k_3\bar{x}r^6 + p_1\left(r^2 + 2\bar{x}^2\right) + 2p_2\bar{x}\bar{y}\\ \delta y &= k_1\bar{y}r^2 + k_2\bar{y}r^4 + k_3\bar{y}r^6 + p_2\left(r^2 + 2\bar{y}^2\right) + 2p_1\bar{x}\bar{y}\end{aligned} \tag{2.19}$$

2.4.2 Exterior orientation parameters

The camera pose, also referred to as exterior orientation parameters (EOPs) of an image (X_c, Y_c, Z_c, ω, φ, κ) are determined through photogrammetric space resection, which requires image observations of at least three a priori GCPs (ground control points), which are points in the object space with known coordinates. The resulting EOPs are expressed in the same coordinate system as the GCPs.

Traditionally, GCPs were generally classified as horizontal control (the position of the point in object space is known with respect to a horizontal datum) or vertical control (the elevation of the point is known with respect to a vertical datum). Separate classifications of horizontal and vertical control have resulted because of differences in horizontal and vertical datums, and because of differences in surveying techniques for establishing horizontal and vertical control. However, currently both the horizontal and vertical coordinates of a GCP are often known. For accurate results, the images of GCPs must be sharp, well defined, and positively identified on the images. They must also be even distributed throughout the image and located at favourable locations on the image. Specifically, the images of the control points should ideally form a large equilateral triangle. A detailed explanation of photogrammetric space resection is provided in Section 6.5.

3 FUNDAMENTALS OF UAV NAVIGATION

Autopilots use onboard navigation sensors and kinematic modelling to determine the position and attitude (i.e., pose) of the UAV. A proven solution is an integrated navigation system that combines Global Navigation Satellite Systems (GNSS) and an Inertial Measurement Unit (IMU) (Chatfield, 1997). Additional sensors such as magnetometers are often used to augment the navigation performance. The following chapter explains the fundamentals of UAV navigation, specifically detail is provided about the autopilot technologies that are used on most common UAV platforms. The chapter focuses on the sensors and software used to process the sensor data to provide a navigation solution.

3.1 Kinematic modelling

Kinematic modelling is the determination of a rigid body's trajectory from measurements relative to a reference coordinate frame. A rigid body is a body with finite dimensions, which maintains the property that the relative positions of all its points, defined in a coordinate frame within the body, remain the same under rotation and translation (Goldstein, 1980). For example, the Paparazzi (wiki.paparazziuav.org) autopilot's unknown state vector, given in Equation (3.1), includes the position in the mapping frame $\mathbf{r}^{\mathbf{M}} = \begin{bmatrix} X & Y & Z \end{bmatrix}^T$, the body velocity $\mathbf{v}^{\mathbf{b}} = \begin{bmatrix} u & v & w \end{bmatrix}^T$, the quaternion representation of the attitude $\mathbf{Q}_{\mathbf{M}}^{\mathbf{b}} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T$, which is the rotation from the mapping frame to the body frame, and the gyroscope biases $\mathbf{b}_{\mathbf{g}} = \begin{bmatrix} b_{g_x} & b_{g_y} & b_{g_z} \end{bmatrix}^T$. These parameters are determined by solving the following system of differential equations.

(3.1)

$$\begin{bmatrix} \dot{\mathbf{r}}^{\mathbf{M}} \\ \dot{\mathbf{v}}^{\mathbf{b}} \\ \dot{\mathbf{Q}}_{\mathbf{M}}^{\mathbf{b}} \\ \dot{\mathbf{b}}_{\mathbf{g}} \end{bmatrix} = \begin{bmatrix} \left(\mathbf{R}_{\mathbf{M}}^{\mathbf{b}}\right)^{\mathbf{T}} \mathbf{v}^{\mathbf{b}} \\ \mathbf{a}^{\mathbf{b}} + \mathbf{R}_{\mathbf{M}}^{\mathbf{b}} \mathbf{g}^{\mathbf{M}} - (\mathbf{\Omega x}) \mathbf{v}^{\mathbf{b}} \\ (\mathbf{\Omega_q x}) \mathbf{Q}_{\mathbf{M}}^{\mathbf{b}} \\ \mathbf{0} \end{bmatrix}$$

where $\mathbf{\Omega x}$ and $\mathbf{\Omega_q x}$ are the skew symmetric matrices shown in the following equations:

(3.2)

$$\mathbf{\Omega x} = \begin{bmatrix} 0 & -\hat{\omega}_z & \hat{\omega}_y \\ \hat{\omega}_z & 0 & -\hat{\omega}_x \\ -\hat{\omega}_y & \hat{\omega}_x & 0 \end{bmatrix}$$

(3.3)

$$\mathbf{\Omega_q x} = \frac{1}{2}\begin{bmatrix} 0 & -\hat{\omega}_x & -\hat{\omega}_y & -\hat{\omega}_z \\ \hat{\omega}_x & 0 & \hat{\omega}_z & \hat{\omega}_y \\ \hat{\omega}_y & -\hat{\omega}_z & 0 & \hat{\omega}_x \\ \hat{\omega}_z & \hat{\omega}_y & -\hat{\omega}_x & 0 \end{bmatrix}$$

where $\hat{\omega}_x, \hat{\omega}_y, \hat{\omega}_z$ are the angular rate measurements from the gyroscope, corrected for gyroscope bias using:

(3.4)

$$\begin{bmatrix} \hat{\omega}_x \\ \hat{\omega}_y \\ \hat{\omega}_z \end{bmatrix} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} - \begin{bmatrix} b_{gx} \\ b_{gy} \\ b_{gz} \end{bmatrix}$$

To solve the system, the IMU accelerations $\mathbf{a^b}$, and angular rates $\boldsymbol{\omega}^{\mathbf{b}}$, are needed as well as the gravity vector $\mathbf{g^M} = [0 \quad 0 \quad g]^T$, where g is 9.81 m/s^2. The quaternion method to represent attitude is preferred over both Euler angle and the direction cosine methods, as it offers accurate and efficient computation methods without singularities. Kong (2000) described a quaternion approach of the INS algorithm for low-cost IMUs. APPENDIX A provides a description of quaternions and the transformations between quaternions Euler angles, and the DCM matrix. Finally, the gyroscope biases $\mathbf{b_g}$ are included in the state to correct the gyroscope measurements. They are modelled as random constants, an unpredictable random quantity with a constant value and covariance (Jekeli, 2001).

3.2 Inertial measurement unit (IMU)

The Inertial Measurement Unit (IMU) consists of a triad of accelerometers and a triad of gyroscopes. Gyroscopes are used to sense angular velocity $\boldsymbol{\omega}^{\mathbf{b}} = \begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^T$, which describes the rotation of the b-frame with respect to the i-frame, coordinated in the b-frame. The i-frame is an inertial reference frame in the Newtonian sense, meaning it is not accelerating nor rotating. These measurements are integrated to provide orientation changes of the body relative to its initial orientation. Although gyroscopes alone are capable of accurately estimating orientation, they are prone to bias and random walk errors resulting from the integration of wide-band noise. Further, the bias stability of the gyro decreases as its cost decreases. Bias fluctuations cause the time-dependent attitude and positional errors to increase without bound and quickly exceed the accuracy specifications for many trajectory determination applications. This is the case with the low-cost MEMS gyroscope being used, thus an augmentation system is required to measure attitude and frequently update the INS state. Accelerometers are used to sense specific force $\mathbf{a}^{\mathbf{b}} = \begin{bmatrix}a_x & a_y & a_z\end{bmatrix}^T$, which describes the specific force of the b-frame relative to the i-frame coordinated in the b-frame. These measurements are used for two purposes. Firstly, they are used to estimate drift-free roll and pitch in the body frame to aid the gyroscopic attitude. Secondly, they are used to derive body accelerations which, after double integration with respect to time, give position differences relative to an initial position. However, this double integration of acceleration data will quickly increase the positional error without bound. Similar to the attitude, frequent updating from external sources, such as GNSS, is needed to bound the error growth. The next section describes the augmentation systems for both attitude and positioning determination.

The most accurate IMUs use Ring Laser Gyroscopes (RLG), however they are the most expensive. Fibre Optic Gyroscopes (FOG) are often preferred because they are less expensive and offer comparable accuracies. Finally, Micro Electro-Mechanical Systems (MEMS) gyroscopes are the least expensive; however, their accuracies are not suitable for many applications. Widely used GNSS/IMU navigation systems include the Applanix APX-15 UAV and AP20 systems (Applanix, 2017), which use a MEMS IMU, or the POS LV 200 systems which use an RLG IMU, while IGI TERRAcontrol uses a FOG IMU (IGI, 2017).

3.3 Sensor augmentation

The first section describes the position augmentation system used to limit the accelerometer-derived position's error growth. The second section describes the attitude augmentation system used to limit the gyroscope-derived attitude's error growth. The Kalman filtering algorithm used to optimally estimate the navigation solution is then detailed.

3.3.1 Position augmentation

As mentioned in the previous section, position is estimated through double-integration of the IMU accelerometer measurements. Double-integration of the accelerometer measurement errors causes the error in position to increase without bound. Position estimates from other sensors are used to mitigate this drift. The following sections describe viable sensors.

3.3.1.1 Global navigation satellite systems (GNSS)

Global navigation satellite systems (GNSS), including Global Positioning System (GPS), GLONASS, and Galileo, are outdoor positioning system that can be used for trajectory determination. GNSS receivers use signals sent by GNSS satellites in orbit around Earth to estimate the geographic location of a GNSS antenna. In order to estimate a GNSS position, the GNSS receiver should receive data from at least four satellites, and increasing the number of satellites will increase the positional accuracy. In addition, the GPS receiver can output a precise clock signal, 1 pulse per seconds (PPS), that can be used for synchronization of the system components.

No other positioning technology offers the same accuracy, long term stability, and flexibility at the same cost. Consequently, GPS is widely accepted and used throughout the world. However, GPS is prone to cycle slips, signal blockage, and multipath, which significantly reduces the GPS

data quality. The accuracy is also subject to environmental conditions, such as poor satellite geometry, atmospheric effects and signal multipath. The models that relate the position and velocity with GPS measurements are well known (Hofmann-Wellenhof et al., 2001).
The most widely used autopilots (Table 1.1) use commercial-grade single frequency (L1) GPS receivers. However, specialized hardware, specifically dual-frequency, or geodetic-grade GPS or GNSS receivers, may be used to determine the UAV position more accurately. Many companies offer such receivers, such as Trimble, Topcon, Leica, and NovAtel. The receiver operates in differential mode relative to a local base station; however, a global DGPS service such as OmniSTAR or national monitoring networks (such as CORS) are replacing the use of local base stations.

Ellum and El-Sheimy (2002) list several modes of Real-time Kinematic (RTK) GPS operation that are potentially applicable to a low-cost UAV, their accuracies are provided in Table 3.1. Precise point positioning (PPP) (Zumberge et al., 1997), an enhanced single point positioning technique, is also a viable option because a GPS base station is not required to collect simultaneous observations. The accuracy estimates for PPP are provided by Bisnath and Gao (2008).

Table 3.1. GPS operation modes (10 km baseline).

GPS solution	**Position accuracy**	
	Horizontal (2D RMS)	**Vertical (RMS)**
L1 carrier-phase RTK (float ambiguities)	0.18 m	0.25 m
L1/L2 carrier-phase RTK (fixed ambiguities)	0.03 m	0.05 m
L1 and L1/L2 post-mission kinematic	0.02 m	0.03 m
L1 real-time PPP (no base)	0.50 m	0.70 m
L1/L2 real-time PPP (no base)	0.10 m	0.15 m

Dual-frequency receivers with firmware supporting RTK-GPS are very expensive compared to single-frequency receivers. However, they have the advantage of much shorter times for ambiguity resolution.

For the PPP solution, error corrections are broadcasted from globally acting services, like the International GNSS Service (IGS, 2017) (GNSS satellite orbits and clocks errors), or regional

GNSS service providers (ionosphere and troposphere delays) to mobile PPP-enabled receivers through an Internet link (e.g., cellular network). By eliminating the base station, PPP saves time, resources and data volumes, and does not limit the high accuracy region to a certain area. However, an initialization time is required: Less than 5 minutes for 50 cm accuracy, 30-60 minutes for 5 cm accuracy, and 24 hours to 0.5 cm accuracy (Seepersad and Bisnath, 2013). Coordinate convergence periods have been reported to take up to 2 hours. Ambiguity resolution using undifferenced GPS observations is also an issue because of the presence of code and phase biases in the ambiguity estimates (Collins et al., 2010).

The L1 carrier-phase Real-Time Kinematic (RTK) GPS solution is the most applicable option for low-cost UAV applications, because the hardware is portable and low-cost, and it provides the accurate positioning in real-time, thus ensuring a high quality solution at the image exposure times. The pseudorange and carrier phase observations are logged for post-mission processing to further improve the positional accuracy. With a single-frequency receiver, at least a few minutes are necessary to obtain a first fixed solution. Takasu (2009) developed an open-source RTK-GPS software package called RTKLIB and obtained cm-level positioning accuracy from L1 RTK-GPS. For carrier-based relative positioning with a short length baseline (less than 15 km) between the kinematic rover *r*, and static base-station *s*, Equations (3.5) and (3.6) are the measurement models for double differenced carrier-phase Φ_{rs}^{ij} and pseudorange P_{rs}^{ij}. In these equations, satellite and receiver clock-biases, and atmospheric effects are eliminated by through double-differencing.

$$\Phi_{rs}^{ij} = \rho_{rs}^{ij} + \lambda\left(B_{rs}^{i} - B_{rs}^{j}\right) + \varepsilon_{\Phi} \tag{3.5}$$

$$P_{rs}^{ij} = \rho_{rs}^{ij} + \varepsilon_{P} \tag{3.6}$$

$$\rho_{r}^{i} = \left\|\mathbf{r}_{r} - \mathbf{r}^{i}\right\|, \rho_{s}^{i} = \left\|\mathbf{r}_{s} - \mathbf{r}^{i}\right\| \tag{3.7}$$

where $(\ )^{ij}$ and $(\ )_{rb}$ represent single-difference between satellites *i* and *j*, and receivers *r* and *s*, respectively. λ is the carrier wavelength, ρ is the geometric range, B_{rs}^{i} is the single-difference carrier-phase float ambiguity between receiver *r* and *s* for satellite *i*, and ε is the measurement error of these observables. The positioning algorithm uses L1 pseudorange and carrier-phase measurements to solve for the unknown state vector in an Extended Kalman Filter (EKF):

$$\mathbf{x} = \left(\mathbf{r}_r^T, \mathbf{B}_{L1}^T\right)^T \tag{3.8}$$

$$\mathbf{B}_{L1} = \left(B_{rs}^1, B_{rs}^2, \ldots, B_{rs}^m\right)^T \tag{3.9}$$

where $\mathbf{r}_r$ is the rover antenna's position in the ECEF frame, and $\mathbf{B}_{L1}$ are the single-differenced L1 float ambiguities. The positioning accuracy is increased by resolving the float carrier-phase ambiguities into their true integer values using the well-known LAMBDA algorithm (Teunissen, 2005), its extension MLAMBDA (Chang et al., 2005) is also used to improve the computational efficiency. The integer ambiguities are then used to re-estimate the rover's position.

The combination of the two complementary measuring systems offers a number of advantages. The IMU provides dense and precise short-term positional data, while GPS provides accurate positioning with long-term stability.

3.3.1.2 Auxiliary sensors

The following sensors may also be used to augment the position solution of a UAV's navigation system. However they are not as widely used, nor as integral as GNSS sensors.

Barometer

Since atmospheric pressure decreases as the height above sea level increases, a pressure sensor can be used to estimate the UAV's height, or altitude. Most flight controllers take input from both the pressure sensor and GPS altitude to estimate height above sea level. Covering the barometer with a piece of foam diminishes the pressure changes caused by wind over the sensor.

Airspeed sensor

An airspeed sensor is a type of pressure sensor that measures how fast the movement of the air passing the aircraft is by calculating the dynamic and static pressure through the pitot tube. This is commonly used on fixed wing aircraft.

Optical flow sensor

A downward optical flow sensor helps maintain the UAV's position when flying over a suitable textured environment. This sensor is used when flying indoors, or in thick tree cover, when it is not always possible to obtain a reliable GPS signal. Currently optical flow is still an experimental feature for most UAV autopilots.

3.3.2 Attitude augmentation

As mentioned in the previous section, attitude is estimated through integration of the IMU gyroscope measurements. Integration of the gyroscope measurement errors causes the error in attitude to increase without bound. Attitude estimates from other sensors are used to mitigate this drift. The following sections describe viable sensors.

3.3.2.1 Muli-antenna GNSS

Ellum and El-Sheimy (2002) list possible sensors for attitude determination and provide the advantages and disadvantages of each. Attitude determination using a GPS multi-antenna system is a viable choice, where three antennas determine the three attitude angles, or two antennas determine two angles. One drawback to this method is that the accuracy of the attitude is proportional to the baseline distance between the antennas. Szarmes et al. (1997) showed that with a sufficient number of GPS satellites and good satellite geometry, 0.75^o accuracies in attitude are attainable with baselines over half a metre. However, using three antennas is not suitable for a low-cost, low-weight, portable system.

3.3.2.2 Attitude and heading reference system

A more appropriate solution, in terms of size, cost, and power consumption uses the IMU's 3-axis accelerometer to estimate roll and pitch, and a 3-axis magnetometer to measure magnetic heading, this is referred to as an Attitude and Heading Reference System (AHRS). Notably, the magnetometer's heading must be transformed to geographic heading by adding the magnetic declination at that location. The value for magnetic declination is freely available from various sources. Global models are accurate to better than 1^{o}, but regions with denser magnetic observations offer better accuracy (NRCAN, 2017).

Equation (3.10) is the model used by the AHRS to relate attitude, expressed in Euler angles roll φ, pitch θ, and yaw ψ, that rotate the local level frame to the IMU body frame, with the accelerometer and magnetometer measurements (Ozyagcilar, 2011).

$$\begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \tan^{-1}\left(\dfrac{a_y}{-a_z}\right) \\ \tan^{-1}\left(\dfrac{-a_x}{a_y \sin(\varphi) + a_z \cos(\varphi)}\right) \\ \tan^{-1}\left(\dfrac{m_e}{m_n}\right) \end{bmatrix} \tag{3.10}$$

$$\begin{aligned} m_e &= m_y \cos(\varphi) - m_z \sin(\varphi) \\ m_n &= m_x \cos(\theta) + m_y \sin(\varphi)\sin(\theta) + m_z \cos(\varphi)\sin(\theta) \end{aligned} \tag{3.11}$$

where m_x, m_y, m_z are the magnetometer measurements, and a_x, a_y, a_z are the accelerometer measurements.

The attitude from this method, unlike the gyroscope's attitude, does not drift, because there is no integration involved; however, there are several limitations. Firstly, error is propagated from the accelerometer and magnetometer measurements, so the attitude is noisier than the gyroscope's attitude (Figure 3.1). The solution also contains systematic errors, such as accelerometer biases and scaling errors, and magnetometer soft iron and hard iron errors, which are due to nearby magnetic fields from permanent magnets, electric currents, or large iron bodies. Caruso (2000) gives a general introduction to digital compasses and magnetic disturbances. Reported accuracies of low-cost MEMS sensors are 0.02^o to 0.5^o, with accuracy increasing with the cost of the sensor. A magnetometer can determine heading with accuracies up to 0.2^o, however the magnetometer operates at a lower sampling interval than the IMU, so the azimuth data is less dense. Notably, the accuracy of these sensors is dependent on temperature, and the accuracy decreases as the tilt angle increases. However, this error is limited since the roll and pitch for terrestrial vehicles are not likely to exceed $\pm 20^{o.}$ Lastly, the attitude can be determined only if the platform's velocity is constant - the accelerometers observe gravity alone.

The combination of these two complementary measuring systems offers a number of advantages. The gyroscope provides high frequency and precise short-term attitude data, while the accelerometer and magnetometer provide long-term stability.

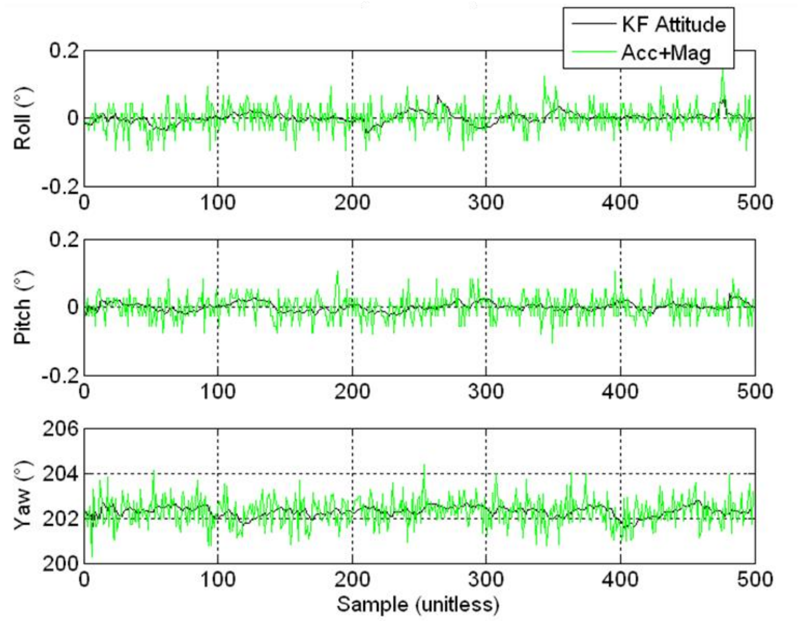


Figure 3.1. Sample data observing static attitude (green is unfiltered, black is filtered).

3.3.3 Optimal state estimation

A Kalman filter (Kalman, 1960) is commonly used in navigation applications to optimally estimate the navigation state, for example, the position, velocity, attitude, and the gyroscope biases. This filter combines the measurements from the GPS, the IMU, and the magnetometer. Hasan et al. (2009) presented a comparative study of various Kalman filter configurations applied to air, land and marine navigation applications. Skaloud (1999) presented and analysed different Kalman filter implementations used to estimate the state vector. The various filter configurations are detailed in the following section.

The Kalman filter includes a system model (also called the dynamic model) that describes how the state propagates in time, and a measurement model that expresses the state as a function of the measurements and updates the state using new measurements. The filter also uses some initial conditions and probabilistic assumptions for the state and observations. In open-source autopilots, such as the Paparazzi (wiki.paparazziuav.org), the system model is based on the differential navigation equations, together with stochastic error models for the gyroscope biases. The measurement model relates the measured GPS position and measured attitude and heading from the accelerometer and magnetometer, respectively, to the state. Due to the nonlinearity of system and measurement models, the well-known EKF (Jazwinski, 1970) is implemented under the assumption that the state behaves as Gaussian random variables. The EKF approximates (or linearizes) the nonlinear functions in the system and measurement models using a first-order Taylor approximation, and thus transforms the nonlinear system into a linear model, similarly to what was done in the previous chapter’s least squares adjustment. The linear discrete EKF (closed-loop estimation) is an optimal recursive data processing algorithm.

Sensor integration can be done in a loosely-coupled (decentralized) or tightly-coupled (centralized) system. In a loosely-coupled system, processed data from the augmentation systems are used to improve the performance of the low-cost INS. Figure 3.2 shows the data flow for a loosely-coupled system. Each system has a local filter to optimally process sensor observations and provide solutions that are subsequently combined in a global filter. In this figure, the local GPS processor generates position and velocity estimates using only GPS observations, the INS processor also estimates the position and velocity from the IMU, the global filter then uses the GPS position and velocity to calibrate and filter the INS positional errors. In a similar manner, another local filter processes accelerometer and magnetometer measurements to provide the attitude solution that corrects the INS attitude errors.

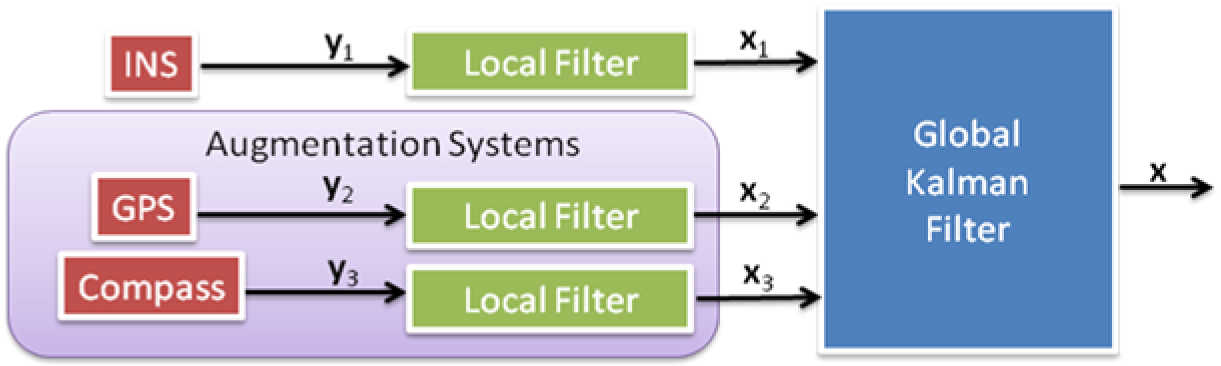


Figure 3.2. Loosely-coupled sensor integration.

In the tightly-coupled system, the sensors are integrated into a single system. The observations from all of the sensors, usually complementary in nature, are processed simultaneously and optimally to enhance the function of individual sensor components. In this case, the global processor uses GPS observations, such as pseudorange and carrier-phase, along with inertial measurements to estimate the navigation parameters (Figure 3.3). This filter is more robust than the loosely-coupled solution because the GPS continues to provide measurement updates regardless of whether or not there is a GPS solution - when there is less than four satellites. However, the centralized filter has a higher computational load and complexity.

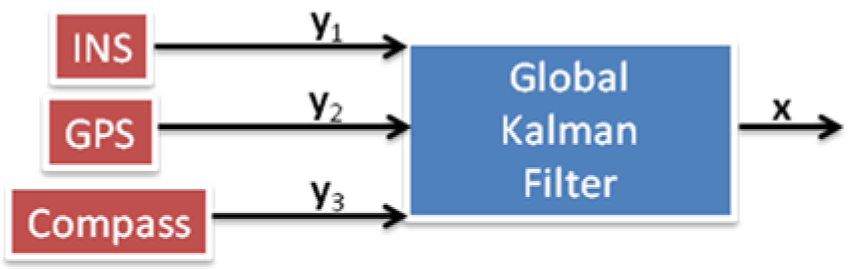


Figure 3.3. Tightly-coupled sensor integration.

It should be noted for completeness, in an ultra-tightly-coupled filter the IMU measurements are used as external inputs to a GPS filter to aid in pre-positioning calculations for faster signal acquisition and in interference rejection during signal tracking (Ahmed et al., 2009).

3.4 Alternative navigation solutions for GNSS-denied environments

There are many literature surveys of existing navigation solutions designed for GNSS-denied environments: (Mautz, 2012), (Doiphode et al., 2016), (Helle et al., 2013), (Torres-Solis et al., 2010), and (Borenstein et al., 1996). Figure 3.4 and Table 3.2 provide a summary of these navigation systems. Detailed descriptions of each measuring principle and technology are provided in (Mautz, 2012).

3.4.1 Interoceptive and exteroceptive measurement sensors

Navigation systems can be divided into two general classes of sensor measurements: Interoceptive and exteroceptive measurement systems (Foxlin, 2005). Interoceptive systems are used for dead reckoning. Measurements are made without reference to any elements external to the sensor. This class includes inertial measurements (Jekeli, 2001) and relative displacement measurements (e.g., from wheel encoders or optical flow sensors (Maye et al., 2006)). Other sensors that measure ranve include ultrasonic sensors (Mautz and Ochieng, 2007), LIDAR (Light Detection and Ranging) (Khoshelham, 2010), and RADAR (Radio Detection and Ranging) (www.symeo.com). Laser scanners, linear CCD (Charge-Coupled Device) sensors, and phased array sensors can be used to measure one-dimensional bearing angles. Imaging sensors, PSD (Position Sensitive Device) sensors, quadcells, and pan/tilt servos can be used to measure two-dimensional bearing angles. Doppler RADAR (Yokoo et al., 2009) and phase-coherent acoustic sensors can be used to detect range rates. TDOA (Time Difference of Arrival) RF or acoustic sensors (Schweinzer and Syafrudin, 2010) can be used to measure range differences. Active source magnetic trackers and electric field trackers (Capps, 2001) can be used to measure dipole field components.

Exteroceptive measurements provide measurements between the sensor and elements external to it. Within this class, some measurements are relative to the reference frame ("earth referenced" measurements). For example, processed GPS/GNSS signals and altimeters can be used to measure Cartesian positions and velocities. Magnetic compasses and gyrocompasses can be used to measure headings. Magnetometers and gravitometers can be used to measure fields that vary by location in the environment. Such measurements are made with respect to physical properties of the environment.

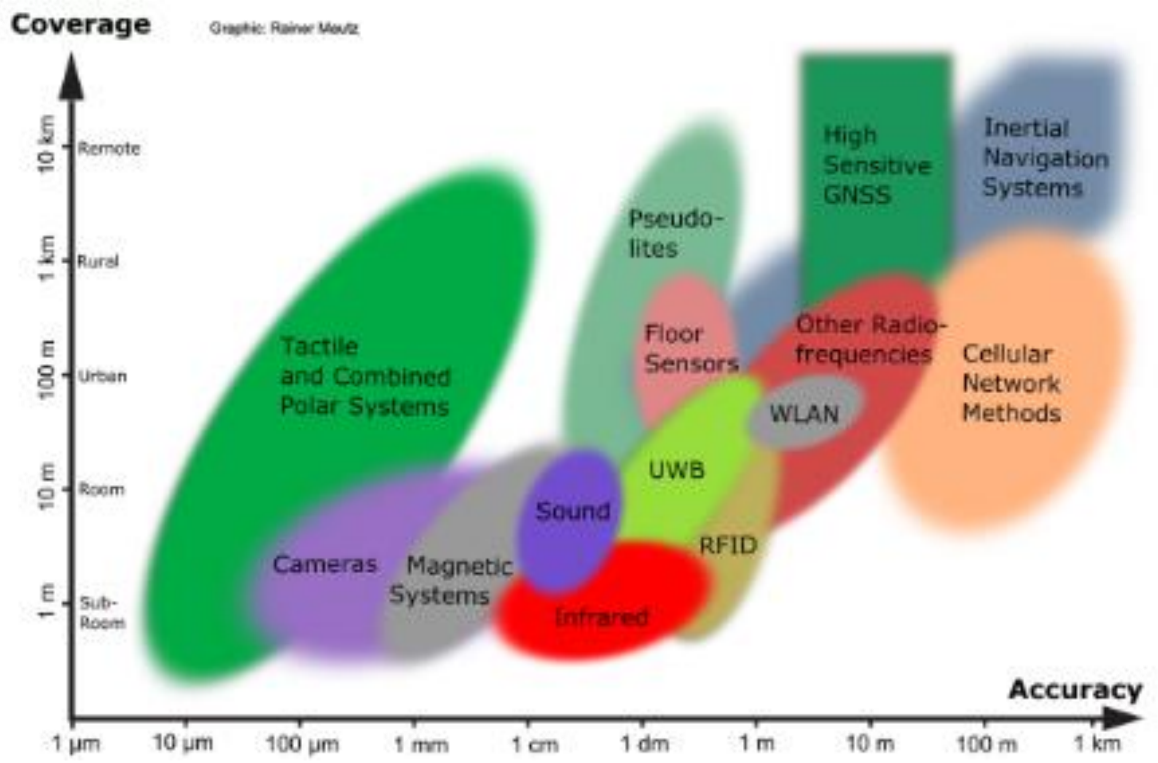


Figure 3.4. Overview of indoor technologies in dependence on accuracy and coverage.

Other sensor measurements are related to pairs of sensors and targets that are fixed to the vehicle and targets or sensors fixed in the environment ("map-referenced" measurements). Examples of sensors include cameras and microphones. Examples of targets include target images (e.g., bar-codes (Mulloni et al., 2009), and retro-reflective targets (Lee and Song, 2007)) and active beacons (Atsuumi and Sano, 2010). It is not necessary that the sensor be fixed to the vehicle and targets fixed in the environment. For example, sensors may also be fixed in the environment such as tracking cameras fixed in a factory environment that track movements of vehicles moving on the factory floor. Targets may be active or passive. An example of an active target is an ultrasound speaker that emits ultrasound signals (Filonenko et al., 2010). An example of a passive target is a circular marking pasted on the ceiling of a room, or a natural landmark such as a corner of a building, or rocks in a terrain.

Table 3.2. Overview of indoor navigation technologies. Coverage refers to ranges of single nodes (Mautz, 2012).

Technology	**Typical accuracy**	**Typical coverage (m)**	**Typical measuring principle**	**Typical application**
Cameras	0.1mm-dm	1-10	Angle measurements from images	Metrology, robot navigation
Infrared	cm-m	1-5	Thermal imaging, active beacon	People detection, tracking
Tactile & polar systems	μm-mm	3-2000	Mechanical, interferometry	Automotive, metrology
Sound	cm	2-10	Distances from time of arrival	Hospitals, tracking
High sensitivity GNSS	10 m	‘global’	Parallel correlation, assistant GPS	Location-based services (LBS)
WLAN/WiFi	m	20-50	Fingerprinting	Pedestrian navigation, LBS
RFID	dm-m	1-50	Proximity detection, fingerprinting	Pedestrian navigation
Ultra-wideband	cm-m	1-50	Body reflection, time of arrival	Robotics, automation
Pseudolites	cm-dm	10-1000	Carrier phase ranging	GNSS challenged pit mines
Other radio frequencies	m	10-1000	Fingerprinting, proximity	Person tracking
Inertial navigation	1%	10-100	Dead reckoning	Pedestrian navigation
Magnetic systems	mm-cm	1-20	Fingerprinting and ranging	Hospitals, mines
Infrastructure systems	cm-m	building	Fingerprinting, capacitance	Ambient assisted living

4 CAMERA-BASED LOCALIZATION

Among the localization solutions presented in the previous chapter, the consensus is that camera-based localization solutions have the most potential to serve the mass market. Low-cost cameras achieve accuracy levels between tens of micrometres and decimetres. The covered areas of the systems reviewed range from 4 m^2 to large room sizes, but can be scaled arbitrarily. High update rates of typically more than 10 Hz allow for kinematic applications such as precision-navigation, real-time mapping and pose estimation.

DeSouza and Kak (2002) describe visual localization approaches where a camera is the only or the main sensor. They categorize the literature into three classes (Figure 4.1): 1) Map-based, 2) map-building, and 3) mapless.

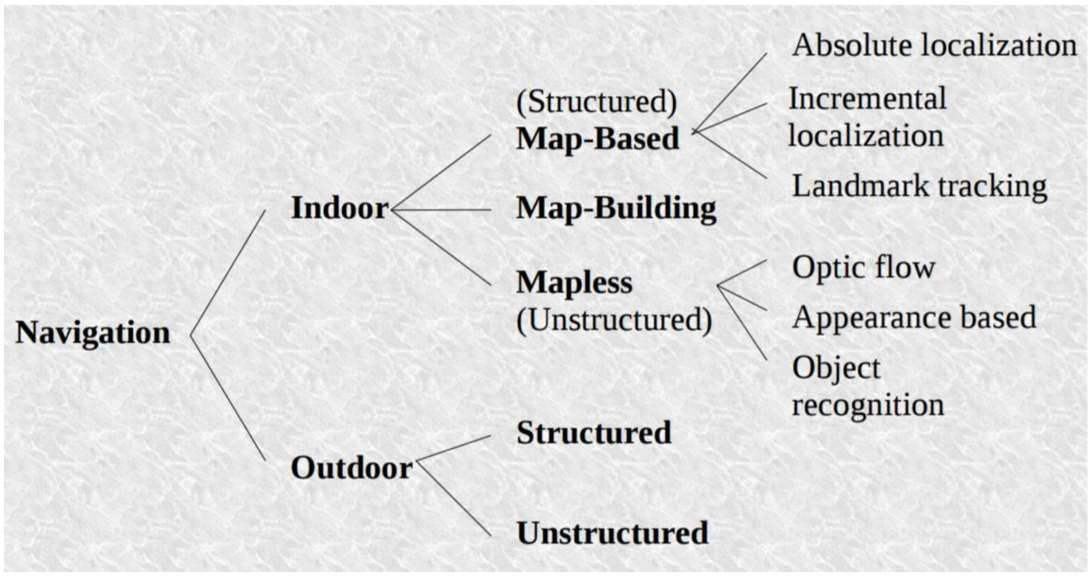


Figure 4.1. Types of camera-based localization solutions (DeSouza and Kak, 2002).

4.1 Map-based localization

Map-based localization assumes a model of the environment is available. These models may contain different degrees of detail, varying from a complete 3D CAD model of the environment to a simple graph of interconnections or interrelationships between the elements in the environment. Since the central idea of map-based localization is to provide, directly or indirectly, a sequence of landmarks expected to be found during navigation, the task of the vision system is then to search and identify the landmarks observed in an image. Once they are identified, the sensor pose can be estimated by using the provided map by matching the observation (image) against the expectation (landmark description in the database). The computations involved in vision-based localization can be divided into the following four steps (Borenstein et al., 1996):

1) Acquire sensory information. For vision-based localization, this involves acquiring and digitizing camera images.
2) Detect landmarks. Usually this involves extracting edges, smoothing, filtering, and segmenting regions on the basis of differences in gray levels, colour, depth, or motion.
3) Establish matches between observation and the map. In this step, the system tries to identify the observed landmarks by searching in the database for possible matches according to some measurement criteria.
4) Calculate camera pose. Once a match (or a set of matches) is obtained, the system calculates the camera pose as a function of the observed landmarks and their positions in the database.

The third step above—establishing matches between observation and expectation—is the most difficult, as the main challenge is achieving real-time capability. Often, this step requires a search that can usually be constrained by prior knowledge about the landmarks and by any bounds that can be placed on the uncertainties in the pose of the sensor. The key advantage of these methods is that there is no requirement for installation of local infrastructure such as deployment of sensor beacons or targets whose positions need to be surveyed beforehand.

For the identification of image correspondences the computational load is particularly high (Mautz, 2012). Different solutions have been proposed to this map-to-image matching problem, Treiber (2010), Chin and Dyer (1986), and Besl and Jain (1985) provide a comprehensive survey. Map-based localization is divided into three categories: 1) incremental localization, 2) absolute localization, and 3) localization derived from landmark tracking. The following sections explain each category.

4.1.1 Map-based incremental localization

In a large number of practical situations, the initial position of the camera is known at least approximately, from the previous frame's camera pose or the autopilot's navigation system for example. In such cases, the localization algorithm tracks the uncertainties in the pose as it traverses the environment and, when the uncertainties exceed a bound, uses its sensors for a new fix on its pose. This problem can be handled by either directly estimating the pose or by estimating the displacement from an image to the next one. By determining correspondences between the 2D visual features and the 3D representation of the object, the 3D rigid transformation between the camera and the object can be computed. Notably, certain factors may affect the visual features and tracking performance across the image sequence. These factors can be illumination changes, background clutter, occlusion of some parts of the object in the image, image noise, image blur, etc. In order to properly address the problem, several issues specific to pose estimation by frame-by-frame tracking should be considered:

- It is assumed that the camera motion between successive frames, expressed in the world coordinate system, is small. This assumption permits the application of various local estimation frameworks (Section 4.1.1.1).
- Given prior knowledge on the object and its representation, attention has to be paid on the type of visual features which can be tracked between two successive frames, and on the way the correspondences between these features and the representation of the object can be determined (Section 4.1.1.2).
- The robustness of feature tracking regarding the factors affecting the visual consistency of the features from one image to the next (Section 4.1.1.3).

4.1.1.1 Pose estimation process

Estimating the 3D transformation between the camera and a map from one image to the next image, can be accomplished using two different approaches: a deterministic approach and a probabilistic approach. Probabilistic techniques are the preferred approach for representing and updating the pose uncertainties as the camera moves. Kalman filters (Bar-Shalom and Li, 1993) have been widely used for tracking and pose estimation of 3D rigid objects (Kosaka and Kak (1992), Gennery (1992), Harris (1992), Koller et al. (1993), and Yoon et al. (2008)). More recently, particle filters (Isard and Blake, 1998) have been used for visual 3D tracking (Klein and Murray (2006), Teulière et al. (2010), Choi and Christensen (2012)). For such methods, a set of hypotheses on the camera pose is propagated with respect to a dynamic model. The pose is then estimated by evaluating the likelihood of the hypotheses in the image. Deterministic, non-linear minimization techniques, such as Gauss-Newton or Levenberg-Marquardt, have also been proposed (Lowe (1992), Drummond and Cipolla (2002), Tamadazte et al., (2010)). The method presented by Marchand and Chaumette (2002), Comport et al. (2003), and Comport et al. (2006) turn the minimization problem into an equivalent visual servoing problem by introducing the Virtual Visual Servoing framework. Others have studied a combination of filtering and deterministic approaches (Teulière et al. (2010), Choi and Christensen (2012)).

Both deterministic and probabilistic methods have their strengths and weaknesses. Iterative, non-linear techniques such as Gauss-Newton or Levenberg-Marquardt are fast (requiring only a few iterations) and accurate, but need to be initialized properly to avoid convergence to local minima. Alternatively, Kalman filtering, which are also fast, use a motion model to predict the state. The motion model stabilizes the estimation and accounts for noisy measurements and underdetermined systems of equations. However, oversimplifying the motion model can result in lag errors. Further, measurement and state noise parameters need to be properly determined and tuned. Finally, the considered visual features and observations can be highly non-linear with respect to the pose parameters, making the computation of the Extended Kalman filter unstable. Particle filters, by representing the state by a set of weighted hypotheses can instead deal with highly non-linear problems and more general distributions of both states and observations. But particle filters may suffer from heavy computational costs (Petit, 2014).

4.1.1.2 Visual features

Whatever the optimization approach, a large set of visual features describing the object and tracked across successive frames has been studied, and could be divided into two categories: local geometrical features and global template-based features.

4.1.1.2.1 Local geometrical features

A common approach of 3D object tracking and pose estimation is to rely on edge features, for instance based on the knowledge of the CAD 3D model of the target. Edge features offer a good invariance to illumination changes and image noise, and are particularly suitable with poorly textured environments. The objective is to estimate the pose that aligns the edge points generated from the projection of the 3D model with their corresponding edge points extracted from the image. Various edge-based geometrical features have been proposed to compute the distance metric to minimize. An early approach (Brown, 1971) chose the point-to-point Euclidean distance. Alternatively, Wuest and Stricker (2007) used perpendicular distances from the projected model edge to a corresponding image edge (Figure 4.2). Other studies match geometrical primitives such as straight lines or segments, circles, etc., to corresponding primitives extracted in the image. For example, Yoon et al. (2008) group edges extracted in the image into segments which are matched to the projected lines of the CAD 3D model. The matching can be based on the Mahalanobis distance of line segment attributes, such as the midpoint, the orientation and the length of the segment (Koller et al., 1993). Once matches have been found, a global error function, computed according to the Malahobnis distance between the projected segments and the extracted ones is also minimized to retrieve the pose.

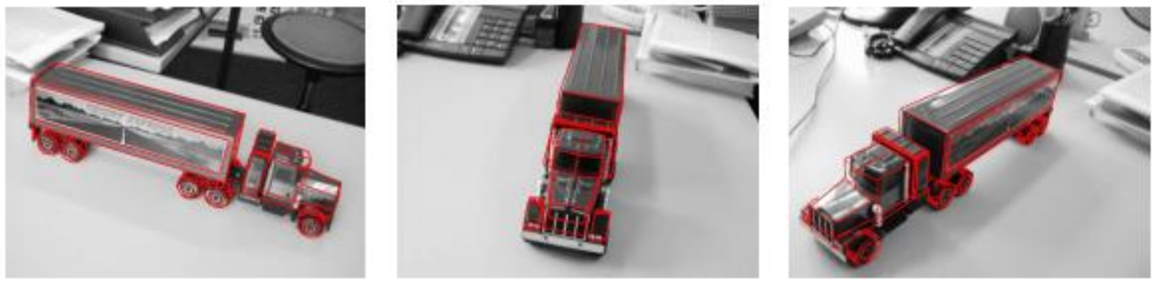


Figure 4.2. Example of edge-based tracking using a toy truck (Wuest and Stricker, 2007).

4.1.1.2.2 Interest point features

Instead of edges, other approaches rely on the detection and tracking or matching of interest points. It relies on matching individual features across images. These features can be automatically detected using feature point detection techniques, for instance the popular Harris corner detector (Harris and Stephens, 1988). These features can then be represented by local patches around the detected point. Matching a point from a frame to a corresponding point in the next frame can be performed through searching on a region surrounding the point through a similarity measure like the Zero-mean Normalized Cross Correlation (ZNCC) (Zhang et al., 1995). Alternatively, the Kanade-Lucas-Tomasi (KLT) (Shi and Tomasi, 1994) algorithm estimates translation parameters of a point from one frame to the next. Pose estimation can then be addressed through 3D-2D correspondences between 3D points lying on a known 3D model of the target, since the 3D coordinates of an image point can be retrieved by back projecting to the 3D model (Petit, 2014).

4.1.1.2.3 Template-based features

Template-based features estimate the displacement of the feature from one frame to the next based on a similarity measure between pixel intensities of the reference template and the image. To account for changes in pixel intensity due to the projection of 3D points to the image, the similarity criteria is often minimized with respect to a considered transformation, through a 2D transformation of the template in the image. For example, Benhimane and Malis (2004) minimized the Sum of Squared Difference (SSD) similarity metric to estimate the homography transformation. Delabarre and Marchand (2013) minimize the Mutual Information similarity metric to estimate a 3D transformation. Optical flow, the 2D motion of pixels lying on the object in the image, has also been used in 3D tracking (Pressigout and Marchand, 2004).

4.1.1.2.4 Hybrid methods

In order to cope with advantages and drawbacks of different types of features, some researches have focused on combining them. Different ways of handling the integration can be considered. Some studies propose a sequential integration of edge and texture features, Marchand et al. (1999) computed dominant motion to provide a prediction of the projected edges in the image, improving edge-based registration. Brox et al. (2006) uses optical flow of pixels lying on the projected object to initialize a maximization process of the separation between object and background along the object contours using colour or luminance probabilities, similar to Prisacariu and Ried (2012).

4.1.1.3 Robust estimation

As previously mentioned, illumination changes and image noise or blur can affect the extraction of some visual features and their matching processes, leading to outliers that can impact the quality of the pose estimation process. Some statistical tools have been used with the objective of reducing the sensitivity of the process to outliers.

4.1.1.3.1 RANSAC

The Random Sample Consensus (RANSAC) method (Fischler and Bolles, 1981) is a general iterative method to estimate parameters of a model from a set of observed data which consists in "inliers" and "outliers". Fischler and Bolles (1981) demonstrated its use in pose estimation. Triplets of 2D-3D point correspondences are randomly selected and processed to compute the resulting set of pose hypotheses. For each pose, all the 3D points are re-projected in the image and the ones which are sufficiently close to their corresponding 2D points in the image are considered as inliers and the pose with the highest number of inliers is chosen as the actual pose. Choi and Christensen (2012) applied this technique for model-based tracking. Armstrong and Zimmerman (1995) used RANSAC for 2D-3D line correspondences.

4.1.1.3.2 M-estimators

M-estimators also perform iterative minimization of an objective function involving measurements, with respect to some parameters. The influence of potential outliers in the observations on the estimation process is reduced by assigning adaptive weights to the observations involved in the objective function. Wuest and Sticker (2007) applied an M-estimators for pose estimation.

4.1.2 Map-based absolute localization

Absolute localization is to be contrasted with incremental localization in which it is assumed that the location of the camera is known approximately at the beginning of a navigation session and that the goal is to refine the pose parameters. In incremental localization, an expectation view is generated using the approximately known pose of the camera. A more precise camera pose estimate is obtained when data from the expectation view are associated with the camera data. Since in absolute localization the initial camera pose is unknown, the navigation system must find a match between the observations and the expectations from the entire database. For example, Petit (2014) generated synthetic views on a view sphere centred on each 3D model (Figure 4.3). Among the literature that addresses this problem, two main categories could be distinguished: global template matching approaches, and local features or descriptor based approaches.

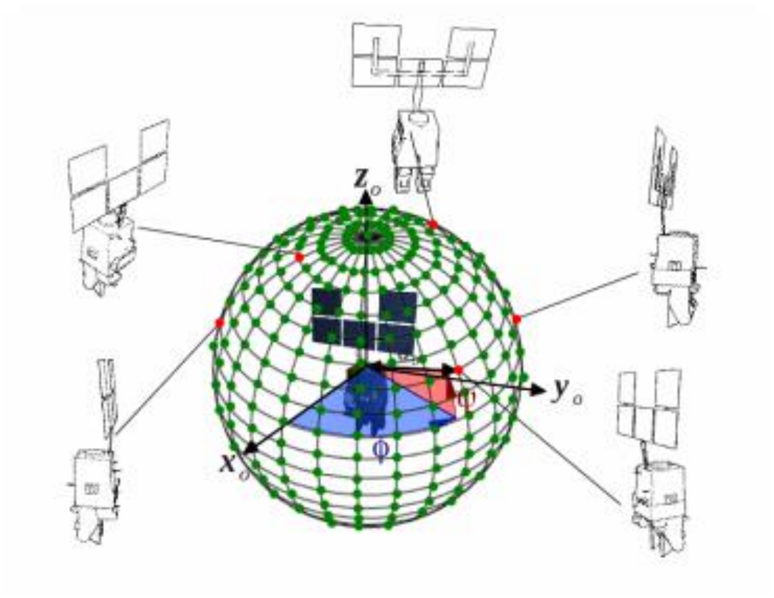


Figure 4.3. Generation of synthetic views on a view sphere centred on the 3D model at regularly sampled viewpoints (Petit, 2014).

4.1.2.1 Global template matching

A training set of images or views of the object is first acquired from many different poses. These views are commonly called templates. At runtime, a similarity measure between the input image and the templates is computed. The template with the highest or lowest score, depending on the similarity measure, is selected as the best match, which then retrieves the camera pose with respect to the object. This very early approach of absolute localization has the advantage of being simple but suffers from two major weaknesses: its computational costs and its sensitivity to appearance changes due to illumination conditions, occlusions or viewpoint changes. For decades, many researches have thus focused on efficiently acquiring, organizing and matching the templates. Common methods of 3D object recognition based on global template matching use natural training templates of the object. Some of them consider appearance ((Ozuysal et al., 2007), (Hinterstoisser et al., 2010), (Gu and Ren, 2010)), or shape ((Holzer et al., 2009), (Payet and Todorovic, 2011)) to represent the object.

4.1.2.2 Local feature-based approaches

These approaches aim at recognizing local or region features from training images or views, and matching them with 2D features in the image. Knowing the 3D relationship between the recognized features, using the 3D model or stored spatial relationships, some methods use the resulting 2D-3D correspondences to directly estimate the pose. Others use the matches in a voting process over the pose parameters space in order to determine the most likely viewpoint and image transformation and finally the pose (Wolfson and Rigoutsos, 1988). Though these features can handle variations in certain conditions, i.e., scale, affine transformations, and illumination changes, a challenge remains in the invariance to viewpoint changes. Example of approaches based on recognizing local features described by descriptors extracted from natural training images of the object include SIFT (Lowe, 2004) or SURF (Bay et al., 2008), contour descriptors (Ferrari et al., 2008), and region descriptors such as HOG (Dalal and Triggs, 2005). The online recognition phase can then provide pose estimates, through a pose computation step based on 2D-3D point correspondences with the 3D model (Ozuysal et al., 2010), using a voting process method (Rodrigues et al., 2016).

4.1.2.3 Approaches using a 3D model

Templates are generated using a 3D CAD model of the environment. Liebelt (2008) extracted SURF local featured from photorealistic rendered views of the 3D model. Different viewing conditions accounted for variations in viewpoint and illumination. Matching was performed through Hough style voting. 3D positions of each SURF feature provided the 2D-3D correspondence required for pose estimation. Edge features and gradient orientations are preferred over feature-based approaches because they are invariant to illumination changes. Matching edge features (Canny, 1986) between model templates and the input image using Hausdorff (Olson and Huttenlocher, 1997) and Chamfer (Shotton et al., 2005) measures has been proposed. However, these measures are sensitive to occlusions. Ulrich (2009) used normalized dot product to match template and image gradient vectors. Pyramid resolution levels for both the templates and input images were used in a coarse-to-fine search over the similarity transformation parameters (translation, scale, and rotation). To optimize the search, views can be clustered into a hierarchical view graph. K-mean clustering (Gravilla (1999), Hinterstoisser (2010)) was initially suggested, but Affinity Propagation (Frey (2007), Reinbacher, (2010)) has been shown to perform better on large sets of data. Jung et al. (2016) demonstrated the efficiency of geometric hashing (Wolfson and Rigoutsos, 1997) in model-to-image matching.

Stark et al. (2010) matched edge information between model templates and the input image by using the Shape Context descriptor (Belongie et al, 2002), Dalal and Triggs (2005) used the HOG feature, and Hinterstoisser et al. (2010) achieved real-time performance using the Dominant Orientation Template (DOT) feature.

On account of the uncertainties associated with the observations, it is possible for the same set of observations to match multiple expectations. The resulting ambiguities in localization may be resolved by methods such as: Markov localization (Thrun, 2000), partially observable Markov processes (Simmons and Koenig, 1995), Monte Carlo localization (Isard and Blake (1998), Dellaert et al. (1999)), multiple hypothesis Kalman filtering based on a mixture of Gaussians (Cox, 1994), using intervals for representing uncertainties (Atiya and Hager (1993), and Krotko (1989)), and by deterministic triangulation (Sugihara, 1988).

4.1.3 Localization derived from target tracking

Optical positioning systems that rely entirely on natural features in images lack robustness, in particular under conditions with varying illumination. In order to increase robustness and improve accuracy of reference points, coded markers may be used for systems with demanding requirements for positioning. The markers serve three purposes: a) simplification of automatic detection of corresponding points, b) establishing the system scale, c) distinction and identification of targets by using a unique code for each marker.

Localization may be performed by target tracking when both the approximate pose of the camera and the identity of the target seen in the camera image are known and can be tracked. Common types of targets include concentric rings, barcodes or patterns consisting of coloured dots (Figure 4.4). There are retro-reflective and non-reflective versions. Target tracking is commonly carried out by using simple template matching. The cameras are usually mounted on the platform so that they look sideways at the walls where the landmarks are mounted or down at the floor where a special tape may be glued (Tsumura, 1986). When cameras are not mounted sideways (Kabuka and Arena, 1987), properties of the shapes used for the targets allow simple algorithms to be used for the localization. Sideways-mounted cameras simplify the problem of target detection by eliminating scale and perspective effects. However, they also constrain the freedom regarding where the camera can move.

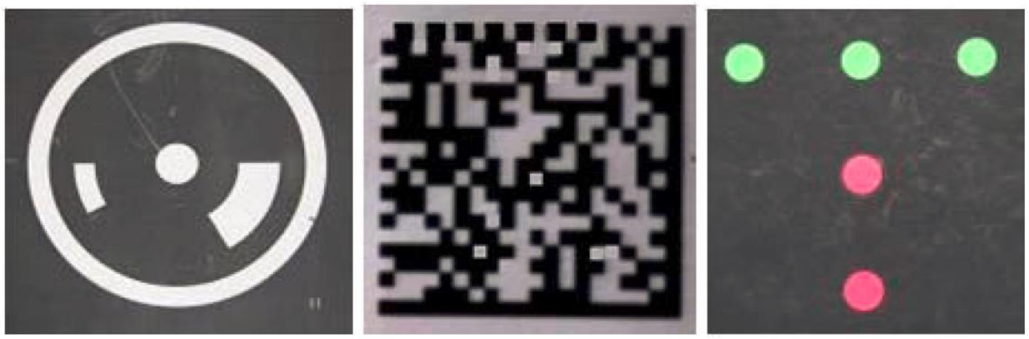


Figure 4.4. Three examples of coded targets used for point identification and camera calibration (Mautz, 2012).

For some applications, mounting targets is undesirable or not feasible. Optionally, infrared light can be projected to attain unobtrusiveness to the user. In contrast to systems relying only on natural image features, the detection of projected reference point or patterns is facilitated due to distinct colour, shape and brightness of the projected features. The principle of an inverse camera (or active triangulation) can be used where the central light projection replaces the optical path of a camera. The main disadvantage of active light based systems is that camera and light source require direct view on the same surface (Mautz, 2012).

4.2 Map-building localization

The vision-based localization approaches discussed so far have all required a map (or a model) of the environment. However, model descriptions are not always easy to generate, especially if one also has to provide metric information. Therefore, many researchers have proposed automated or semi-automated robotic platforms that could explore their environment and build an internal representation of it. Simultaneous localization and mapping (SLAM) and concurrent mapping and localization (CML) techniques search for strategies to explore, map and self-localize simultaneously in unknown environments. Davison and Kita (2001) discuss sequential localization and map building, review the state of the art and expose the future directions that this trend should take. Furthermore, they present a tutorial of first-order relative position uncertainty propagation and software to perform sequential mapping and localization. Further detail about SLAM is provided in Sections 5.2 and 5.3.

4.3 Mapless localization

In this category, navigation is achieved without any prior description of the environment. Contrasting with map-building localization, in mapless localization, no maps are ever created. The camera motions are determined by observing and extracting relevant information about the elements in the environment. These elements can be the walls, objects such as desks, doorways, etc. It is not necessary that absolute (or even relative) positions of these elements of the environment be known. However, navigation can only be carried out with respect to these elements. Of the techniques that have been tried for this, the prominent ones are: optical flow-based, appearance-based, and object recognition-based.

4.3.1 Optical flow

Optical-flow-based solutions estimate the motion of objects or features within a sequence of images. Researchers compute optical flow mostly using (or improving) techniques originating from Horn and Schunck (1981), and Lucas and Kanade (1981). Green et al. (2003) developed a UAV platform called Closed Quarter Aerial Robot (CQAR) that flies into buildings, takes off and lands controlled by an insect-inspired optical flow-based system. Later, Green et al. (2004) extended his optical flow-based navigation system for UAVs to fly in near ground environments such as tunnels, caves, inside buildings or among trees. Hrabar et al. (2005) presented in a navigation technique for UAVs to fly in between urban canyons. The authors combined stereo forward-looking cameras for obstacle avoidance and two sideways-looking cameras for stable canyon navigation. The system projected 3D stereo data onto a 2D map and performs a region growing process to extract obstacles. The UAV tried to balance the optical flow from both sides, moving to the direction of larger optical flow magnitude. The main disadvantage of optical flow based navigation systems is that the positional error accumulates without bound, continuously degrading the accuracy as time passes.

4.3.2 Appearance-based

Appearance-based matching techniques are based on the storage of images in a previous recording phase. These images are then used as templates. The camera navigates in the environment by matching the current viewed frame with the stored templates. Examples of these approaches are: Matsumoto et al. (1996) VSRR (View Sequenced Route Representation), Jones et al. (1997), and Ohno et al.'s (1996) solution is similar but faster than Jones': it only uses vertical lines from templates and on-line images to do the matching. It saves memory and computation time. Nakamura and Asada (1995) provide behaviour-based approaches to vision-based localization.

4.3.3 Object recognition

Techniques for tracking moving elements (corners, lines, object outlines or specific regions) in a video sequence have become robust enough so as to be useful for navigation. The tracking task is commonly divided into two sub-problems (Trucco and Plakas, 2006): First, motion detection, which, given a feature to be tracked, identifies a region in the next frame where it is likely to find such a feature, and second, feature matching, by which the feature tracked is identified within the identified region. Although video tracking and mobile robot navigation belong to separate research communities, some authors claim to bridge them to motivate the development of new navigation strategies. Some authors centre their research in detecting and tracking the ground space across consecutive images, and steering the robot towards free space. Pears and Liang (2001) use homographies to track ground plane corners in indoor environments, with a new navigation algorithm called H-based Tracker. Liang and Pears (2002) extend this work using also homographies to calculate height of tracked features or obstacles above the ground plane during the navigation process. The accuracy of the navigation strategy must be a strategic point in aerial motion where the speed is high, the processing time must be reduced and the tracking process needs to be more accurate. Ollero et al. (2004) maintain an estimate of the homography matrix to compensate the UAV motion and detect objects. Saeedi et al. (2006) use stereo vision to navigate in unstructured indoor/outdoor environments. Detected corner features are positioned in 3D and tracked using normalized mean-squared differences and correlation measurements. Supporting vision information with GNSS data in outdoor environments is another possibility of increasing reliability in position estimation. Mejias et al. (2005) combined a feature tracking algorithm with GNSS positioning in the navigation system of the autonomous helicopter AVATAR. The vision process combined image segmentation and binarization to identify pre-defined features, such as house windows, and a Kalman filter-based algorithm to match and track these windows. The Scale Invariant Feature Transform (SIFT) method, developed by Lowe (1999), stands out among other image feature or relevant points detection techniques, and has become a method commonly used in landmark detection applications. SIFT-based methods extract features that are invariant to image scaling, rotation, and illumination or camera view-point changes. During the UAV navigation process, SIFT features are used as landmarks to be tracked for navigation, global localization (Se et al., 2005) and vision-based SLAM (Se et al., 2002).

4.4 Outdoor localization in structured environments

In general, outdoor navigation in structured environments uses road-following techniques. Road-following implies an ability to recognize the lines that separate the lanes or separate the road from the berm, the texture of the road surface, and the adjoining surfaces, etc. In systems that carry out road following, the models of the environment are usually simple, containing only information such as vanishing points, road and lane widths, etc. Road-following for outdoor robots can be like hallway-following for indoor robots, except for the problems caused by shadows, changing illumination conditions, changing colours, etc. (Desouza and Kak, 2002).

4.5 Unstructured outdoor localization

An outdoor environment with no regular properties that could be perceived and tracked for navigation may be referred to as unstructured environment. In such cases, the vision system can make use of at most a generic characterization of the possible obstacles in the environment. Unstructured environments arise in cross-country navigation as, for example, in planetary (lunar/martian-like) terrain navigation (Matthies et al., 2007). In these applications, the robot is often wanders around, exploring the vicinity of the robot without a specific goal. However, in other applications, the task to be performed requires that the robot follow a specific path to a goal position. In those cases, a map of the traversed areas has to be created and a localization algorithm has to be implemented in order to carry out goal-driven navigation.

5 COMBINING MAP-BASED AND MAP-BULDING INCREMENTAL LOCALIZATION

This chapter describes in more detail the state-of-the art in map-based and map-building incremental localization. Map-based localization focuses on estimating the camera's position and orientation (pose) when a map of the environment is available. A branch of this field of research is called model-based tracking when the map is a 3D model. Common applications of model-based tracking include augmented reality (AR), robot navigation (e.g., automotive), and robotic object manipulation. Lahdenoja et al. (2015) reviewed the advances of monocular, model-based tracking from 2005 to 2014. Lepetit and Fu (2005) reviewed the status of monocular, model-based, rigid body tracking from 1990 to 2005.

Section 5.2 describes two state-of-the-art map-building localization techniques: First, a method called Structure-from-Motion (SfM) uses an RGB camera to perform simultaneous localization and mapping (SLAM). SfM has been widely used by UAVs in outdoor environments. However, the data processing strategy fails in indoor environments because tracking is often lost due to rapid camera motions, small sensor footprints, and prevalent homogeneous textures in the environment. Algorithms like RGB-D SLAM (Endres et al., 2014) have shown more success in indoor environments. An RGB-D sensor is used to perform SLAM. The addition of 3D depth sensor data in the feature tracking algorithm improves the tracking performance by generating a larger number of features, especially in homogeneous area. Further, computational efficiency is increased by directly measuring 3D geometry instead of computing it photogrammetrically.

Map-based localization fails when the map is not present in the camera's field of view or when landmarks cannot be matched between the image and the map. Failures can occur often if the environment has few features to matches or the map has few features to match, i.e., the map has a low Level-of-Detail (LoD). For this reason, the first contribution of this dissertation proposes to "bridge the gaps" between sparse map-based localization updates with (higher frequency) map-building localization updates, i.e., SLAM updates. Further, this dissertation proposes that map-based localization can be used to overcome the weaknesses of map-building localization. If loop-closures are not performed / detected, there will be large drift errors in the map-building localization solution. Additionally, the memory and computational requirements of map-building algorithms increases significantly as the mapped area grows, because of the large amount of data being processed simultaneously. Map-based localization "closes loops" on a pre-existing map, this removes the need in SLAM to maintain the entire map for the purposes of removing drift with loop-closures. Instead the SLAM process can be restarted every time the loop is closed on the pre-existing map (with a map-based localization update). For example, Figure 5.1 is a schematic diagram of a mobile camera traversing from a start position (symbolized by a star) to an end position (symbolized by a triangle). Each frame's estimated position is shown using a circle. The camera pose's accuracy is symbolized by the circle's radius. The unbounded error from SLAM in the absence of loop closures is reduced by the map-based localization updates. In this example, doors appearing in the camera image are matched with doors in the 3D CAD model. The error is reduced by estimating the 3D transformation that aligns the door extracted from the image to the door from the 3D CAD model. Similarly, Tamaazousti et al. (2011) used points from a partial 3D model, and Prisacariu et al. (2013) used shape information from a very coarse 3D model in a SLAM system, to simultaneously reconstruct the whole scene or refine the 3D model, with pose estimation benefiting from this reconstruction.

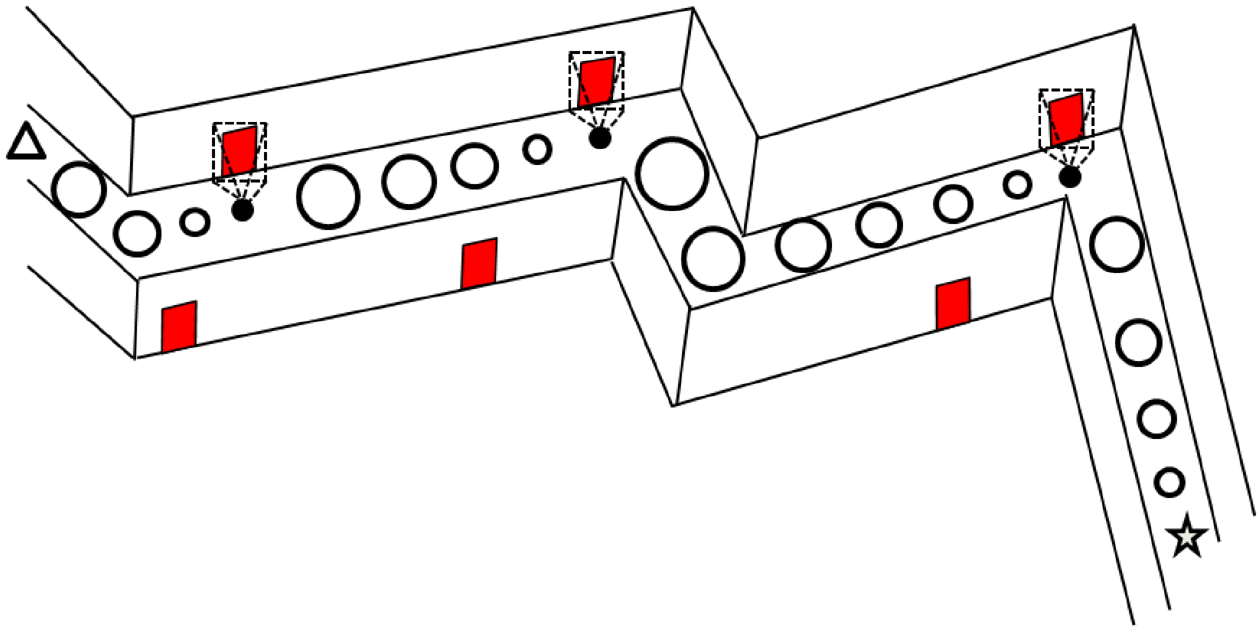


Figure 5.1. Combining map-based and map-building localization techniques.

5.1 Introduction to model-based tracking

A typical workflow of model-based tracking is:

1) In the initialization phase, corresponding points or lines between the image and the 3D model are found either manually by the user or automatically using a map-based absolute localization approach.
2) In the tracking phase, pose is updated each frame by maintaining the correspondences when the camera moves. The pose is resolved from the correspondences, for example using photogrammetric space resection via the collinearity equations.

The following sections provide more detail on each step of this workflow.

5.1.1 Initialization and recovery

Initialization is performed when the tracking starts; the initial pose of the system is estimated without knowledge of the previous states of the system. Pose recovery refers to estimating the pose when tracking is lost. Lahdenoja et al. (2015) and Lepetit and Fu (2005) mention that initialization and recovery is mostly done manually. Automated techniques, referred to as *tracking-by-detection,* fall under the definition of map-based absolute localization (DeSouza et al., 2002). They are divided into two categories:

1) View-based, are edge-based techniques, where edges extracted from current frame are matched with 2D views of the wireframed target object previously obtained from different positions and orientations (Wiedemann and Steger, 2008).
2) Keypoint-based, where keypoints (i.e., image features invariant to scale, viewpoint and illumination changes) extracted from the current frame are matched against a database of keypoints extracted from images of the object taken at different positions and orientations (Skrypnyk and Lowe, 2004).

In both cases, the matches provide 2D-3D correspondences needed for pose estimation. The difficulty in implementing such approaches comes from the fact that the database images and the input ones may have been acquired from very different viewpoints. The so-called wide baseline matching problem becomes a critical issue that must be addressed. A more comprehensive survey of model-based initialization can be found in Euranto et al. (2014).

Lahdenoja et al. (2015) identified that tracking-by-detection techniques commonly use either fiducials (i.e., landmarks or markers) or the object's texture is used. The fourth contribution of this dissertation (Chapter 7) is a map-based absolute localization algorithm that is based solely on natural geometric features.

5.1.2 The tracking phase

In model-based tracking, two categories of features are used for tracking:

1) Edge-based, where camera pose is estimated by matching a wireframe 3D model of an object with the real world image edge information. These methods can be grouped into two categories:
    a. Gradient-based, where model edges are matches with strong gradients in the image, without explicitly extracting the image primitives (i.e., lines or polygons) (Wuest et al., 2005).
    b. Contour-based, where image contours are extracted, such as straight line segments and the model outlines are fit to these image contours (Lowe, 1992).
2) Texture-based techniques rely on information provided by pixels inside the object's projection. These are classified in three sub-categories:
    a. Optical flow, which estimates the relative movement of the object projection onto the image (Basu et al., 1996).
    b. Template matching, which applies a distortion model to a reference image to recover rigid object movement (Jurie and Dhone, 2001);
    c. Keypoint-based, which takes into account localized features in the camera pose estimation (Vacchetti et al., 2004).

Lahdenoja et al. (2015) suggest that the current state-of-the-art, and most widely used method, of model-based tracking is the edge-based methods. Edge-based methods are both computationally efficient, and relatively easy to implement. They are also naturally stable to lighting changes, even for specular materials, which is not necessarily true of texture-based methods. Liu and Huang (1991) also identified some advantages of using lines rather than point features. Firstly, lines are often easier to extract from a noisy image than point features. Secondly, the orientation of a line may be estimated with sub-pixel accuracy. Thirdly, lines have simple mathematical representation. Fourthly, image lines can easily be found in structured indoor and outdoor environments.

5.1.3 Visual Servoing Package (ViSP)

The Visual Servoing Platform's (ViSP) is an open-source implementation of a state-of-the-art model-based tracker. It provides registration techniques that continuously track and align features extracted from the video image and the 3D model of the environment (Marchand and Chaumette, 2002). Amin and Govilkar (2015) provide a comparative study of Augmented Reality SDK's that offer model tracking capabilities. ViSP was chosen over alternative model-based trackers, such as those offered by Vuforia (www.vuforia.com), Wikitude (www.wikitude.com), ARmedia (www.armedia.it), and ARToolkit (www.artoolkit.org) because it tracks geometry and does not rely on the 3D model having texture. ViSP was chosen over the other model-based trackers mentioned in Lahdenoja et al. (2015), for example RAPID (Harris, 1992), because it employs the widely used moving edges tracker (Wuest et al., 2005), (Reitmayr and Drummond, 2006), (Comport et al., 2006), (Choi and Christensen, 2010), (Kyrki and Kragic, 2011), (Seo et al., 2011). Further, ViSP's implementation was professionally executed and made available in an open-source format, which allows users to understand the functionality, and provides the capability to add and improve features. Finally, ViSP is the only model-based tracker available for use with ROS (Robot Operating System) (www.ros.org), a robotics middleware that facilitates system integration. The contributions proposed in this dissertation were implemented and integrated using ROS.

ViSP’s tracking techniques are divided into two classes: feature-based and model-based tracking. The first approach focuses on tracking 2D geometric primitives in the image, such as points (Shi and Tomasi, 1994), straight line segments (Smith et al., 2006), circles or ellipses (Vincze, 2001), or object contours (Blake and Isard, 1998). However, ViSP does not provide a means to match these tracked image features with the 3D model, which is required in the georeferencing process. The second method is more suitable because it explicitly uses a 3D model of the object in the tracking algorithm. Further, a by-product of this tracking method is the 3D camera pose, which is the main objective. This second class of methods usually provides a more robust solution (for example, it handles partial occlusion of the objects, and shadows).

ViSP has demonstrated its capabilities in applications such as augmented reality, visual servoing, medical imaging, and industrial applications (ViSP, 2017). These demonstrations involved terrestrial robots and robotic arms, equipped with cameras, to recognize and manipulate small objects (e.g., boxes, tools, and cups) in cluttered indoor environments (Figure 5.2).

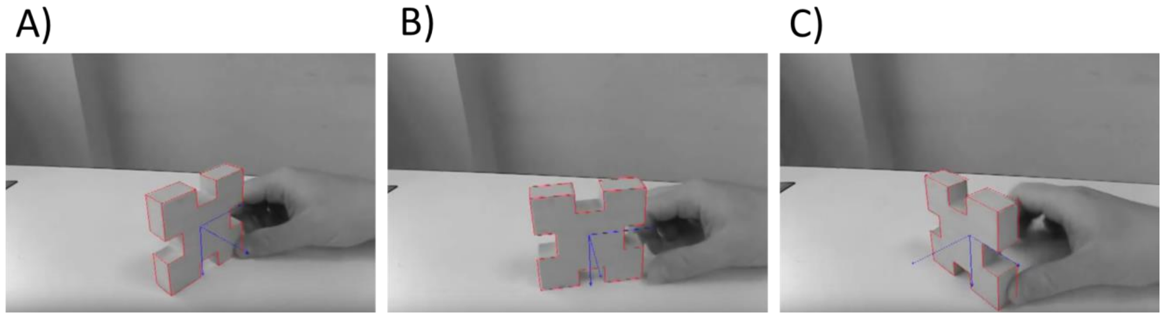


Figure 5.2. Sample frames demonstrating ViSP’s model-based tracker (ViSP, 2013).

The overall workflow is shown in Figure 5.3. Firstly, a moving edges Tracker identifies corresponding features between an image sequence and a 3D model. The algorithm requires the camera pose of the first image from the user. Secondly, Virtual Visual Servoing (VVS) uses the corresponding features to estimate the camera pose in the 3D model’s coordinate system. This camera pose is fed back into the moving edges tracker. To initialize ViSP, the camera pose of the first image frame is estimated via VVS after the user manually selects 4 points minimum in the first image frame that correspond to pre-specified points in the 3D model (Figure 5.4). The following sections explain the algorithms in more detail.

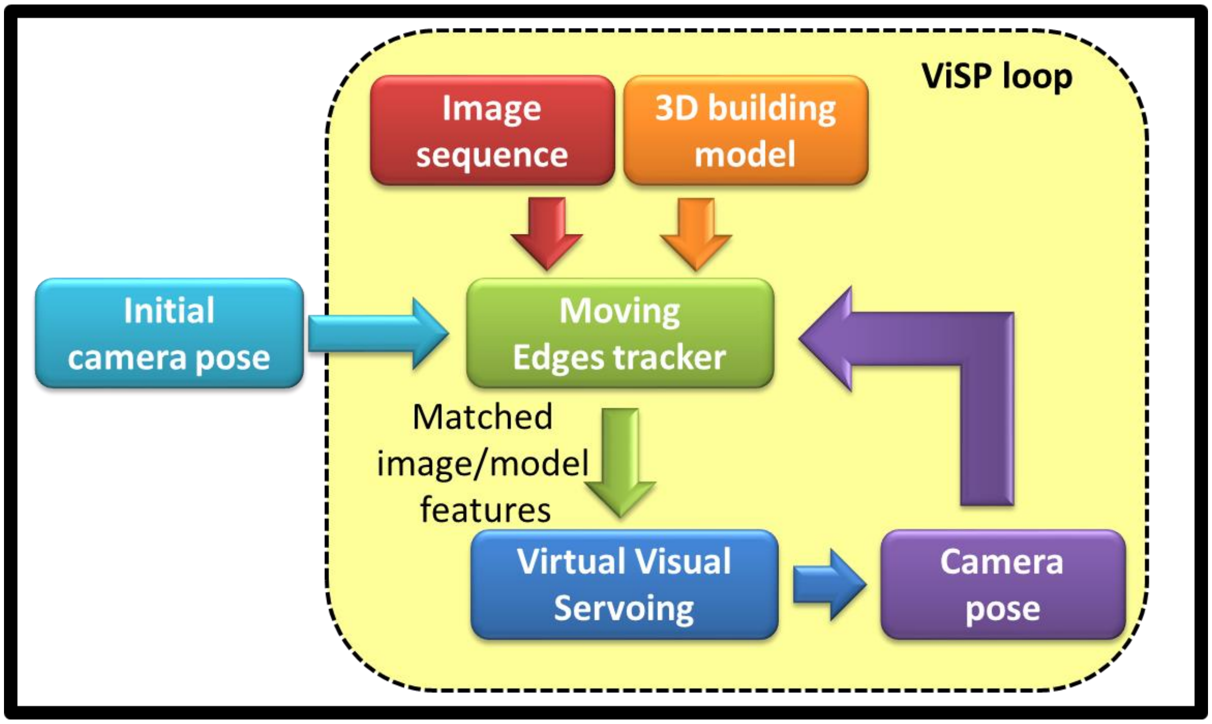


Figure 5.3. ViSP's pose estimation workflow.

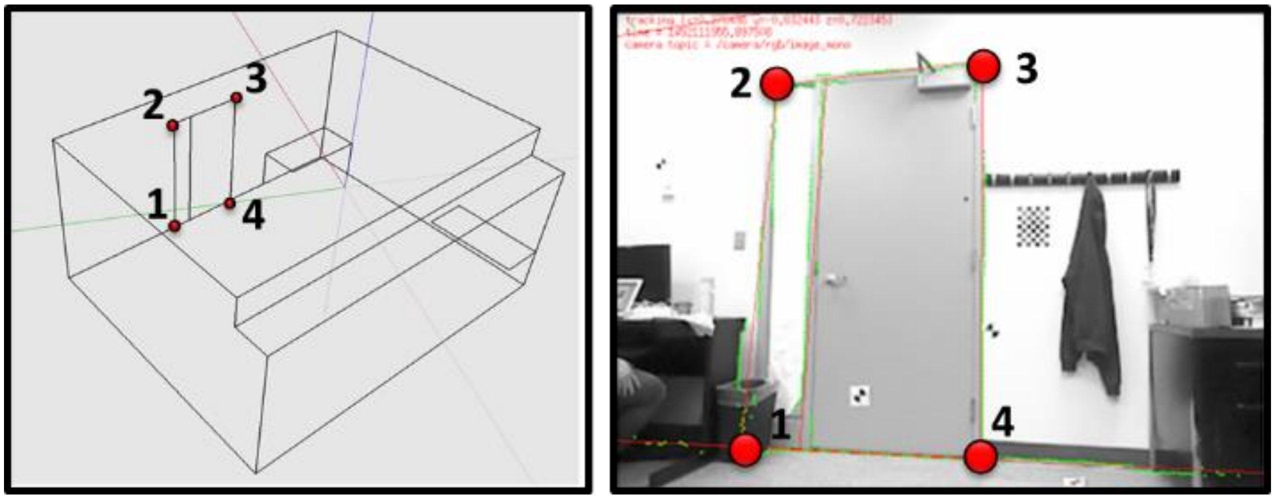
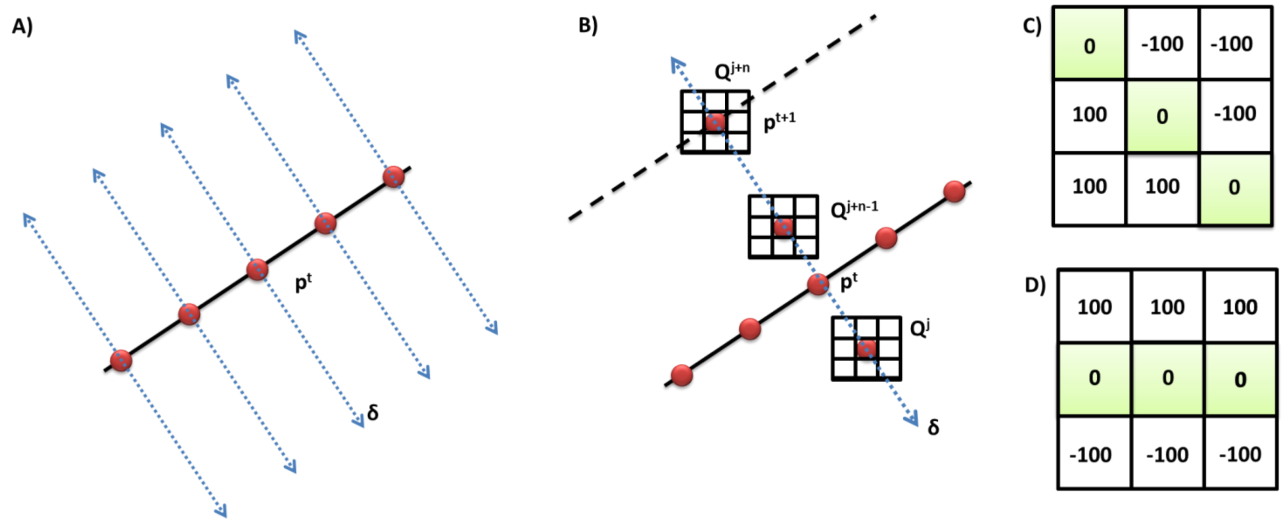


Figure 5.4. Example of ViSP's initialization process. Left) 4 pre-specified 3D points on the wireframe model. Right) The user selects the 4 corresponding points on the image. ViSP uses these corresponding points to estimate the first camera pose.

5.1.3.1 The moving edges tracker

The moving edges algorithm (Boutemy, 1989) matches image edges in the video image frames to the 3D model's edges, which are called model contours in ViSP. The required initial approximation of the camera pose can be provided manually, or by the UAV's autopilot GPS positioning and orientation from the AHRS, if available.

Model contours are sampled at a user specified distance interval (Figure 5.5A). At these sample points (e.g., $\mathbf{p}^t$), a one-dimensional search is performed along the normal direction ($\boldsymbol{\delta}$) of the contour for corresponding image edges (Figure 5.5B). An oriented gradient mask is used to detect edges (e.g., Figure 5.5 C and D). One of the advantages of this method is that it only searches for image edges, which are oriented in the same direction as the model contour. An array of 180 masks is generated off-line that is indexed according to the contour angle. The run-time is limited only by the efficiency of the convolution, which leads to real-time performance (Comport et al., 2003). Line segments are favourable features to track because the choice of the convolution mask is simply made using the slope of the contour line. There are trade-offs to be made between real-time performance and both mask size and search distance.



C)

0	-100	-100
100	0	-100
100	100	0

D)

100	100	100
0	0	0
-100	-100	-100

Figure 5.5. Determining point position in the next image using the oriented gradient algorithm: A) calculating the normal at sample points, B) Sampling along the normal, C)-D) 2 out of the 180 3x3 predetermined masks, C) 180^{o}, D) 45^{o} (Modified from Comport et al., 2003).

The process consists of searching for the corresponding point $\mathbf{p}^{t+1}$ in the image $\mathbf{I}^{t+1}$ for each point $\mathbf{p}^{t}$. A 1D search interval $\{Q_j, j \in [-J, J]\}$ is determined in the direction $\boldsymbol{\delta}$ of the normal to the contour. For each position Q_j lying the direction δ, a mask convolution $\mathbf{M}_\delta$ corresponding to the square root of a log-likelihood ratio ζ_j is computed as a similarity measure. Thus, the new position p^{t+1} is given by:

(5.1)

$$Q^{j*} = \underset{j \in [-J,J]}{\operatorname{argmax}} \; \zeta_j$$

where

(5.2)

$$\zeta_j = \left| I_{\upsilon(Q_j)}^{t+1} * M_\delta + I_{\upsilon(p^t)}^{t} * M_\delta \right|$$

where $\upsilon(.)$ is the neighbourhood of the considered pixel, ViSP's default is a 7x7 pixel mask (Comport et al., 2003).

5.1.3.2 Virtual Visual Servoing

ViSP treats pose estimation as a 2D visual servoing problem as proposed in (Sunareswaran and Behringer, 1998). Once each point's search along its normal vector finds a matching model point via the moving edges tracker, the distance between the two corresponding points is minimized using a non-linear optimization technique called Virtual Visual Servoing (VVS). A control law adjusts a virtual camera's pose to minimize the distances, which are considered as the errors, between the observed data s_d (i.e., the positions of a set of features in the image) and $s(\mathbf{r})$, the positions of the same features computed by forward-projection of the 3D features **P.** For instance in Equation (5.3), ${}^{\mathbf{o}}\mathbf{P}$ are the 3D coordinates of the model's points in the object frame, according to the current extrinsic and intrinsic camera parameters:

(5.3)

$$\Delta = (s(\boldsymbol{r}) - s_d) = \left[pr_\xi(\boldsymbol{r},\, {}^{o}\boldsymbol{P}) - s_d\right]$$

where $pr_\xi(\boldsymbol{r},\, {}^{o}\boldsymbol{P})$ is the projection model according to the intrinsic parameters ξ and camera pose $\mathbf{r}$, expressed in the object frame**.** It is assumed the intrinsic parameters are available, but

VVS can estimate them along with the extrinsic parameters. An iteratively re-weighted least squares (IRLS) implementation of the M-estimator is used to minimize the error Δ. IRLS was chosen over other M-estimators because it is capable of statistically rejecting outliers.

Comport et al. (2003) provides the derivation of ViSP's control law. If the corresponding features are well chosen, there is only one camera pose that allows the minimization to be achieved. Conversely, convergence may not be obtained if the error is too large.

5.1.4 TIN to polygon 3D model

ViSP specifies that a 3D model of the object to track should be represented using VRML (Virtual Reality Modeling Language). The model needs to respect two conditions:

1) The faces of the modelled object have to be oriented so that their normal goes out of the object. The tracker uses the normal to determine if a face is visible.
2) The faces of the model are not systematically modelled by triangles. The lines that appear in the model must match image edges.

Due to the second condition, the 3D building models used in the experiments must to be converted from TIN to 3D polygon models. The algorithm developed to solve this problem is as follows:

1) Region growing that groups connected triangles with parallel normals.
2) Extract the outline of each group to use as the new polygon faces.

The region growing algorithm was implemented as a recursive function. A seed triangle (selected arbitrarily from the TIN model) searches for its neighbouring triangles, that is, triangles that share a side with it, and have parallel normals. The neighbouring triangles are added to the seed's group. Then each neighbour looks for its own neighbours. The function terminates if all the neighbours have been visited or a side does not have a neighbour. For example, the blue triangles in Figure 5.6 belong to one group. Once all of the triangles have been grouped, the outline of each group is determined (the black line in Figure 5.6). Firstly, all of edges that belong to only one triangle are identified, these are the outlining edges. These unshared edges are then ordered so the end of one edge connects to the start of another. The first edge is chosen arbitrarily. Figure 5.7A) shows the 3D TIN model of York University's Lassonde Building. Figure 5.7B) shows the resulting polygon model of the same building.

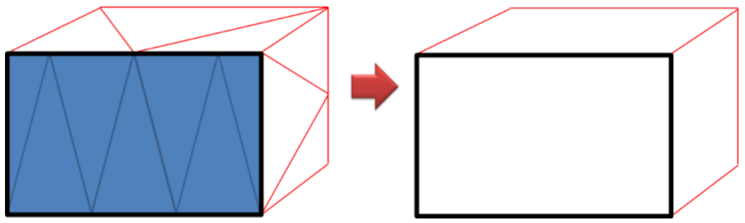


Figure 5.6. An example of region growing and outline detection: The blue triangles belong to a group because one triangle is connected to at least one other triangle with a parallel normal. The outline of the group (black line) consists of the edges that belong only to one triangle.

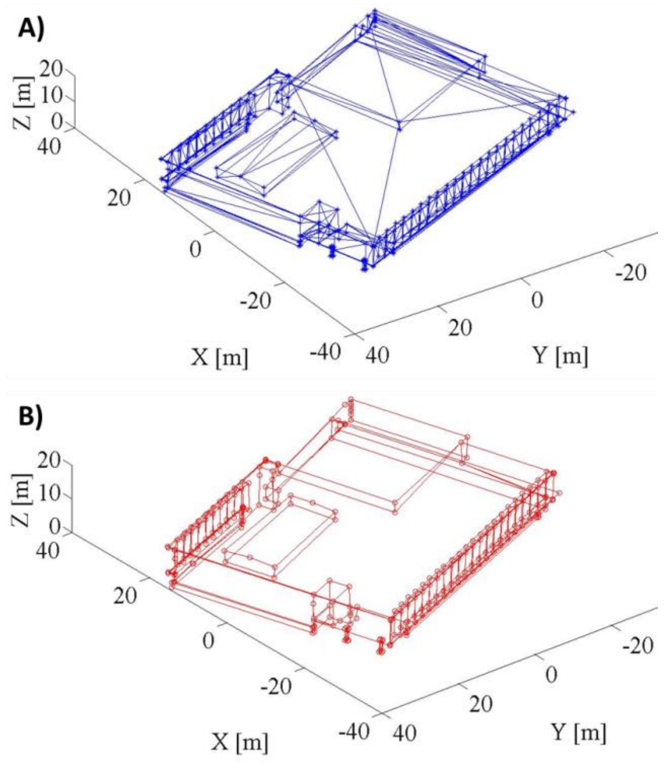


Figure 5.7. Converting TIN building models to polygon models. A) The original TIN model of the York University's Lassonde Building. B) The resulting 3D polygon model of the Lassonde Building.

5.2 Incremental map-building localization using structure from motion

Structure from motion (SfM) is an incremental map-building localization technique. It is built on the concept of photogrammetric triangulation (Gini et al., 2013). Lightweight UAVs and heavy winds cause unstable flight conditions. As a result, vision-based navigation for UAVs must consider larger than average along-track and cross-track overlaps, (e.g., 80% and 60%, respectively), and in turn much more images and processing resources. Tie point generation between successive overlapping images has been performed using many techniques, for example dense multi-view stereopsis (Furukawa and Ponce, 2010) and image matching using SIFT (Lowe, 2004), SURF (Bay et al., 2008), ASIFT (Morel and Yu, 2009), LDAHash (Strecha et al., 2012), and the Semi-Global Matching (SGM) based dense matching SURE (Rothermel et al., 2012) algorithms. The centimetre level spatial resolution achieved by the image sensors on-board the UAV makes it possible to obtain very dense 3D point clouds (Figure 5.8) which are comparable to those generated by laser scanners.

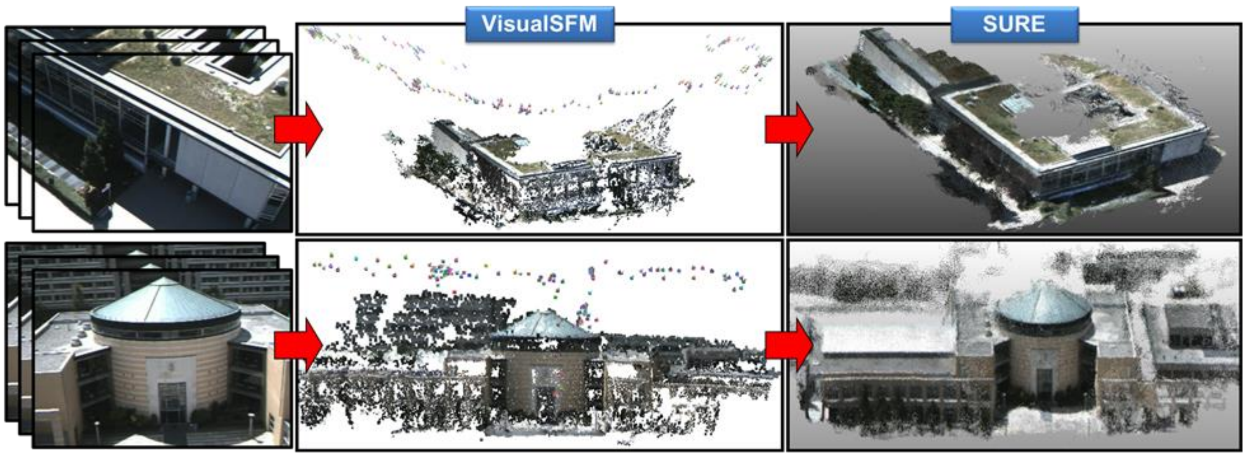


Figure 5.8. Dense point cloud generation from UAV imagery.

Dense multi-image matching is used to generate a 3D point cloud representing the surface of the mapped area from a series of oblique video images. The position and orientation of the image frames are also estimated in the process. The video images captured from the onboard camera are reconstructed based on an incremental Structure from Motion (iSfM) bundle adjustment approach, using VisualSFM (Wu, 2014). Using the SfM approach, the image connectivity of this bundle adjustment solution estimates relative camera positions and orientation elements. Figure 5.9 (left) shows the SfM workflow. The point density is increased using the multi-view stereo approach (Figure 5.9 (right)). A final step is necessary to geo-reference the relative photogrammetric network with the geodetic terrain coordinate system. The Iterative Closest Point (ICP) (Besl and McKay, 1992) procedure based on a 3D similarity transformation may be used to refine the registration between the photogrammetrically derived 3D point cloud with existing 3D building models. The following sections describe each component of the SfM workflow.

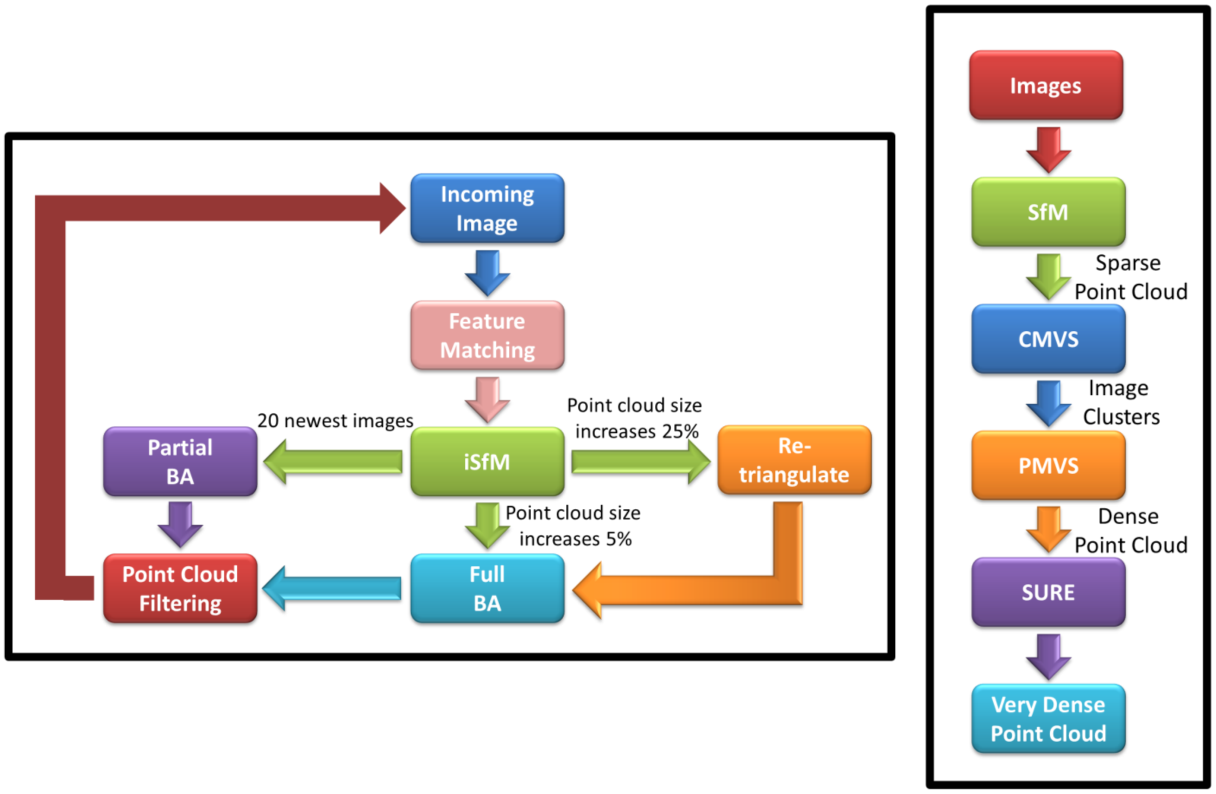


Figure 5.9. SfM software workflow (left) and dense point cloud generation (right).

5.2.1 Incremental Structure from Motion (iSfM)

In iSfM (Wu, 2013), a two-view reconstruction is first determined by triangulating successful feature matches between two images. Incoming images are then repeatedly matched and the 3D model is extended from the two-view reconstruction. One image is added at each iteration. In order alleviate error accumulation, partial bundle adjustments (BAs) can be run using a constant number of recently added images (e.g., about 20) and their associated 3D points. Following the

BA step, filtering removes the points that have large re-projection errors or small triangulation angles. Finally, the next iteration starts or a re-triangulation occurs (Section 5.2.1.2).

The camera pose, or exterior orientation, and the 3D point estimations typically converge quickly during reconstruction, thus full BAs (on all cameras and 3D points) are performed when the size of a model increases by a certain ratio (e.g., about 5%). Although the latter added cameras are optimized by fewer full BAs, there are normally no accuracy problems because full BAs improve less accurate parts. Notably, as the photogrammetric model gets larger, more cameras are added before running a full BA (Wu, 2013). As SfM produces sparse point clouds, multi-view stereo (MVS) algorithms, such as CMVS (Section 5.2.2) and PMVS (Section 5.2.3), may be used to increase the point density. CMVS takes the output from the SfM software and produces image clusters. PMVS runs significantly faster and produces more accurate results with the produced clusters.

5.2.1.1 Image matching

Image matching is one of the most time-consuming steps of SfM. Due to the wide range of viewpoints in a large collection of photos, Wu (2013) states that the majority of image pairs do not match (~75% - 98%). A large portion of matching time is saved by identifying the good pairs robustly and efficiently. For instance, the approximate GNSS tags are used to match images only to the nearby ones (Frahm et al., 2010). Further, pre-emptive feature matching (Wu, 2013) filters correspondence candidates based on the scales of SIFT features (Lowe, 2004), as the chances of correctly matching the top-scale features is higher than matching randomly selected features. Finally, increases in matching speed can be achieved by parallelizing the search with multiple machines (Agarwal et al., 2009) and multiple GPUs (Frahm et al., 2010).

5.2.1.2 Bundle adjustment

A set of measured image feature locations and correspondences are input into a bundle adjustment, with the goal of finding triangulated 3D point positions and camera parameters that minimize the bundle reprojection error (Triggs et al., 1999). This optimization problem is treated as a non-linear least squares problem, where the error is the squared norm of the difference between the observed feature location and the reprojection of the corresponding 3D point on the image plane of the camera. The Levenberg-Marquardt (LM) algorithm (Nocedal and Wright, 2000) is used for solving non-linear least squares problems. Wu et al. (2011) demonstrated a CPU-based BA that is up to 10 times faster, and a GPU-based system that is up to 30 times faster, than the 2011 state-of-the-art BA.

When GPS is not available, the iSfM solution is prone to drift because of the accumulated errors of relative camera poses. The initially estimated poses and even the poses after a partial BA may not be accurate enough and this may result in some correct feature matches failing the quality test. As the drift is attributed mainly to the accumulated loss of correct feature matches, failed feature matches are re-triangulated (Wu, 2013) when the model size increases (e.g., by 25%). After re-triangulating, a full BA and point-filtering is run to improve the reconstruction. This strategy of reducing the drift is analogous to loop-closing.

5.2.2 Clustering Views for Multi-View Stereo (CMVS)

Multi-view stereo (MVS) algorithms aim to correlate measurements from a collection of images to derive 3D surface information. Many MVS algorithms reconstruct a single 3D model by using all the images available simultaneously. As the number of images grows the processing time and memory requirements become infeasible. To solve this problem, subsets of overlapping images are clustered into manageable pieces that are processed in parallel, and the resulting reconstructions are merged (Furukawa et al., 2010). The clustering algorithm is designed to satisfy the following three constraints: 1) redundant images are excluded from the clusters, 2) each cluster is small enough for an MVS reconstruction (a size constraint determined by computational resources), and 3) MVS reconstructions from these clusters result in minimal loss of content and detail compared to that obtained by processing the full image set. Having extracted image clusters, a patch-based MVS software (PMVS) is used to reconstruct 3D points for each cluster independently. CMVS increases the performance of PMVS by removing images that insignificantly and negatively impact the resulting point cloud.

5.2.3 Patch-Based Multi-View Stereo (PMVS)

The Patch-Based Multi-View Stereo (PMVS) algorithm (Furukawa and Ponce, 2010) represents scene surfaces through collections of small oriented 3D rectangular patches (essentially local tangent planes). The algorithm consists of a simple match, expand, and filter procedure:

1) Matching: Features found by Harris and Stephens (1988) and difference-of-Gaussians operators (Lowe, 2004) are first matched within each cluster of pictures, yielding a sparse set of patches associated with salient image regions. A matching patch is considered to be an inlier if the search along the epipolar lines of other images yields low photometric discrepancies (one minus the normalized cross correlation score) in a minimum number of images (e.g., 2 or 3). Given these initial matches, the following two steps are repeated.
2) Expansion: The initial matches are spread to nearby pixels and obtain a dense set of patches.
3) Filtering: Visibility constraints are used to eliminate incorrect matches lying either in front or behind the observed surface.

5.2.4 Photogrammetric Surface Reconstruction

The SURE dense image matching algorithm (Rothermel et al., 2012) densifies a point cloud by using the oriented images generated from the bundle adjustment. These images are first rectified to generate epipolar images and dense disparities are calculated across stereo pairs using Semi-Global Matching (SGM) (Hirschmüller, 2008). Briefly, the SGM algorithm performs the image alignment required to estimate disparities by the maximizing of the mutual information (i.e., minimizing the joint entropy) between two overlapping images. Instead of using the entire image in this calculation (global matching), 16 one-dimensional directional paths are constructed to approximate the image (semi-global matching). 3D points or depth images are then triangulated from the stereo models. Finally, redundant depth measurements are used to remove outliers and increase the accuracy of the depth measurements.

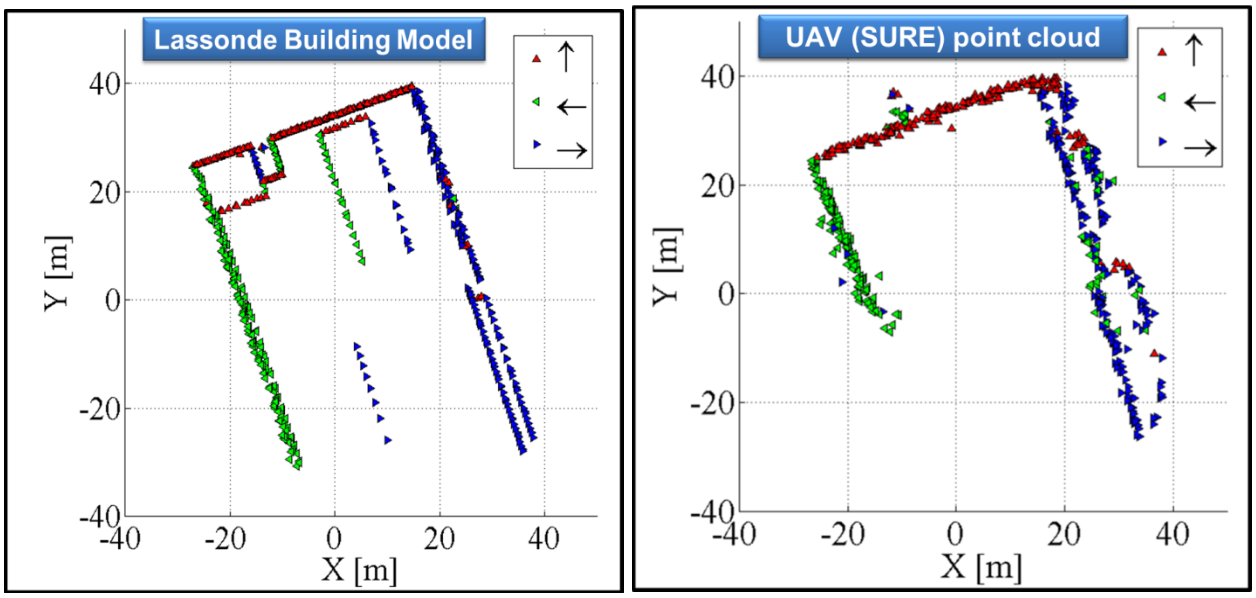


Figure 5.10. Segmentation of the Lassonde Building model (left) and corresponding point cloud generated from the UAV data (right). Points are segmented into three groups based on their normal vector: Group 1 (red triangles): Normals pointing in the positive Y direction ($\pm 45^o$), Group 2 (green triangles): Normals pointing in the negative X direction ($\pm 45^o$), and Group 3 (blue triangles): normal pointing in the positive X direction ($\pm 45^o$).

5.2.5 Iterative Closest Compatible Point (ICCP)

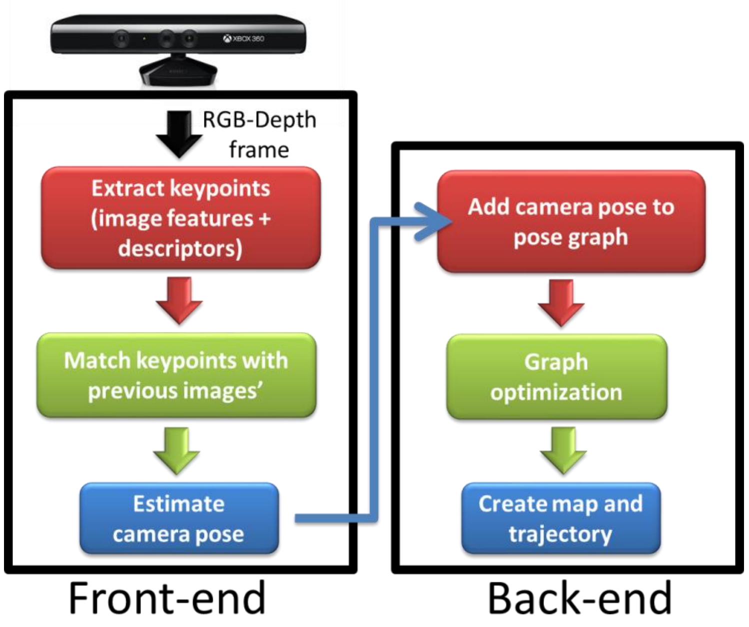
The Iterative Closest Compatible Point (ICCP) algorithm may be used to improve the geodetic position of the point cloud. ICCP, an ICP variant, iteratively refines the rigid-body transformation between the SfM point cloud and a point cloud sampled from the 3D model. Given two point clouds that are roughly aligned, ICP uses all of the points to refine their relative 3D similarity transformation. Correspondence is established by pairing points in one point cloud with the closest points in the other and the relative transformation between the two point clouds is refined by minimizing the points' Euclidean distance error metric. The initial alignment may be provided by a model-based tracker or GPS positioning for example. Several additional heuristics have been developed to address the problem of not knowing the extent of overlap and avoiding false point matches. For example, the Iterative Closest Compatible Point (ICCP) algorithm (Godin et al., 1994) matches points only if their associated feature (colour, normal vector, etc.) are within a given threshold. ICCP can be interpreted as a set of rigidly coupled ICP sub-problems between subsets of mutually compatible points. ICCP is equivalent to ICP when all points have compatible attributes. For example, Figure 5.10 shows the faces of a building model (left) and a point cloud (right) segmented into three classes based on the normal vector. Further, the point-plane (Chen and Medioni, 1992) method improved upon the original point-point ICP as it converges an order of magnitude faster, it is more robust against outliers, and produces more accurate results (Pulli, 1999).

5.3 Incremental map-building localization using RGB-D SLAM

RGB-D SLAM (Endres et al., 2014) follows the general graph-based SLAM approach. The system consists of a front-end module and a back-end module (Figure 5.11). The front-end processes the sensor data, i.e., the sequence of RGB and depth images, to compute the sensor's motion relative to detected landmarks. A landmark is composed of a high-dimensional descriptor vector extracted from the RGB image, such as SIFT (Lowe, 1999) or SURF (Bay et al., 2008) descriptors, and its 3D location relative to the camera pose of the depth image. The relative motion between two image frames is computed via photogrammetric bundle adjustment using landmarks appearing in both images as observations. Identifying a landmark in two images is accomplished by matching landmark descriptors, typically through a nearest neighbour search in the descriptor space.

Continuously applying this pose estimation procedure on consecutive frames provides visual odometry information. However, the individual estimations are noisy, especially when there are few features or when most features are far away, or even beyond the depth sensor's measurement range. Combining several motion estimates, by additionally estimating the transformation to frames other than the direct predecessor, commonly referred to as loop closures, increases accuracy and reduces the drift. Notably, searching for loop closures can become computationally expensive, as the cost grows linearly with the number of candidate frames. Thus RGB-D SLAM employs strategies to efficiently identify potential candidates for frame-to-frame matching.

The back-end of the SLAM system constructs a graph that represents the camera poses (nodes) and the transformations between frames (edges). Optimization of this graph structure is used to obtain a globally optimal solution for the camera trajectory. RGB-D SLAM uses the g2o graph solver (Kümmerle et al., 2011), a general open-source framework for optimizing graph-based nonlinear error functions. RGB-D SLAM outputs a globally consistent 3D model of the perceived environment, represented as a coloured point cloud (Figure 5.12).


RGB-Depth frame
Extract keypoints (image features + descriptors)
Match keypoints with previous images'
Estimate camera pose
Add camera pose to pose graph
Graph optimization
Create map and trajectory
Front-end
Back-end
Figure 5.11. RGB-D SLAM's workflow for pose estimation and map creation.

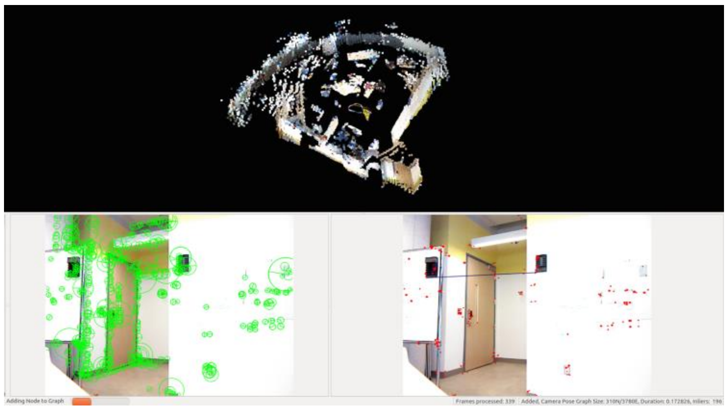


Figure 5.12. Screen capture of RGB-D SLAM. The top window shows the map (RGB point cloud) being created. The bottom right and left windows show the SIFT image feature (red dots) and their scales (green circles), respectively.

5.4 Combining incremental map-based and map-building localization

This dissertation assessed the performance of Visual Servoing Platform's (ViSP) pose estimation algorithm. Experimentation revealed that ViSP failed when tracking was lost, and needed to be manually re-initialized (Section 5.5). This occurred when there was a lack of model features in the camera's field of view, and because of rapid camera motion. The first contribution of this dissertation proposes that the tracking performance of incremental map-based localization (e.g., the ViSP model-based tracker) improves when map-building localization (e.g., RGB-D SLAM) concurrently provides camera pose estimates.

Further, experimenting with stand-alone RGB-D SLAM revealed that drift was often present in g2o's globally optimized trajectory (Section 5.6). There were two reasons for this: Firstly, data were not collected that enabled a loop closure, i.e., not returning to a previously occupied vantage point. Secondly, the loop closure was not detected by RGB-D SLAM. That is, the user is able to configure various parameters that affect the probability of detecting a loop closure, for the reason that reducing this probability increases computational efficiency of the overall system. Since it is infeasible to compare every frame with every other frame, the user is able to specify the number of frame-to-frame comparisons to sequential frames, random frames, and graph neighbour frames. Alternatively, ViSP's pose estimation process can be thought of as loop closing on the 3D wireframe model instead of RGB-D SLAM's map. This allows the RGB-D SLAM's loop closure parameters to be set very low, essentially turning RGB-D SLAM into a computationally efficient visual odometry system, without sacrificing the accuracy provided by large loop closures.

In summary, by integrating the two systems, not only does RGB-D SLAM improve ViSP's tracking performance, but ViSP substitutes RGB-D SLAM's loop closing, with improved runtime performance. The proposed integrated solution is analogous to GPS/INS integration, where ViSP provides a low-frequency and drift-free pose estimate that is registered to a (georeferenced) 3D model, similar to GPS. Complementarily, the RGB-D SLAM visual odometry solution provides a high frequency pose estimate that drifts over time, similar to the behaviour of an INS.

Figure 5.13 shows the strategy for integrating map-based (e.g., model-based tracking (MBT)) and map-building (e.g., SLAM) localization. Firstly, the model-based tracker is initialized by receiving corresponding features between the first image frame(s) and the 3D model. A set of matching features are detected either manually by the user or automatically by using a map-based absolute localization algorithm for example. The MBT uses these corresponding image-model matches to estimate the first camera pose with respect to the model's coordinate system. The SLAM system then begins mapping and tracking the camera pose, and continues until the MBT detects corresponding image and model feature points. The MBT then corrects the camera pose and SLAM is reset at the MBT's provided pose. Upon resetting, SLAM saves each frame, with its corresponding features and camera pose, which were collected since the last reset. Saving this data allows the map to be incrementally built, without burdening the SLAM system by processing too much data simultaneously.

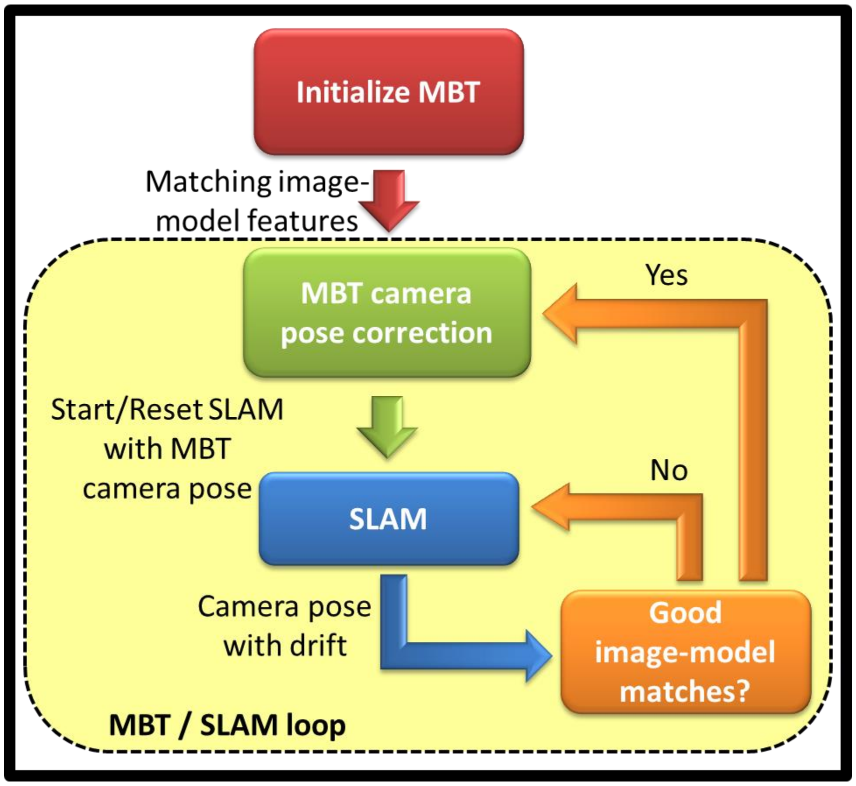

Initialize MBT
Matching image-model features
MBT camera pose correction
Yes
Start/Reset SLAM with MBT camera pose
No
SLAM
Camera pose with drift
Good image-model matches?
MBT / SLAM loop

Figure 5.13. Model-based tracker / SLAM integration workflow.

5.5 Results: A UAV application of state-of-the-art map-based incremental localization

This dissertation first presents results from the application of a state-of-the-art model-based tracker in larger environments, tracking larger, more complex objects. Specifically, ViSP (Visual Servoing Package) was applied in two situations:

1) Tracking the pose of a UAV's camera by tracking the facades of a georeferenced outdoor 3D building model.
2) Tracking the pose of a UAV's RGB camera by tracking a georeferenced indoor 3D model.

Given an initial approximation of the camera pose, ViSP automatically establishes and continuously tracks corresponding features between an image sequence and a 3D wireframe model of the environment. As ViSP has been demonstrated to perform well in small and cluttered indoor environments, this work explores the application of ViSP in mapping large outdoor environments by UAVs, and tracking larger objects (i.e., building models). The presented experiments demonstrate the data obtainable by the UAV, assess ViSP's data processing strategies, and evaluate the performance of the tracker.

The following section provides experimental results from the assessment of ViSP in tracking large, complex objects using an RGB camera onboard a UAV. It is then demonstrated that tracking accuracy and maintenance is improved by integrating RGB-D SLAM into the localization system.

5.5.1 The UAV

This section presents the developed indoor/outdoor navigation system based on the Arducopter quadrotor UAV. The Arducopter is equipped with a Pixhawk autopilot, a small forward-looking 0.3MP camera and an Occipital Structure sensor (www.occipital.com), which is a 0.3 MP depth camera, capable of measuring ranges up to 10 metres ± 10%.

5.5.2 The Pixhawk autopilot

The Pixhawk autopilot comprised of a GPS sensor that provides positioning accuracies of about 3 m, and an Attitude and Heading Reference System (AHRS) that estimates attitude to about 3°. The Pixhawk processes the sensor data in two stages, attitude determination, followed by position determination. Attitude is optimally estimated using a loosely-coupled filtering solution, which integrates the gyroscopic attitude solution with the accelerometer and magnetometer attitude and heading. This system is called the Attitude and Heading Reference System (AHRS). Position estimation is based on single-frequency GPS single point position solutions, loosely-coupled in a Kalman filter with the IMU. As shown in Figure 5.15, the AHRS attitude $\mathbf{R}_M^b$ is used to transform the body accelerations into the mapping frame.

5.5.3 Data collection and processing

In testing, the UAV flew over York University, up to approximately 40 metres above the ground, while its onboard camera focused on buildings, walkways, and trees. Figure 5.16 shows the UAV's flight path for the presented experiment. The 3D virtual building model of York University's Keele Campus (Armenakis and Sohn, 2009) was used as a known environment. The model consists of photorealistic 3D TIN (Triangulated Irregular Network) reconstructions of buildings, trees, and terrain (Figure 5.17). The model was generated from building footprint vector data, Digital Surface Model (DSM) with 0.75 metre ground spacing, corresponding orthophotos at 0.15 metre spatial resolution and terrestrial images. The 3D building models were further refined with airborne LIDAR data having a point density of 1.9 points per square metre (Corral-Soto et al., 2012). This 3D CAD model serves two purposes in the proposed approach. Firstly, it provides the necessary level of detail such that individual buildings can be uniquely identified via ViSP's moving edges algorithm (Boutemy, 1989). Secondly, it provides ground control points to photogrammetrically estimate the camera pose. The geometric accuracy of the building models is in the order of 10 to 40 centimetres.

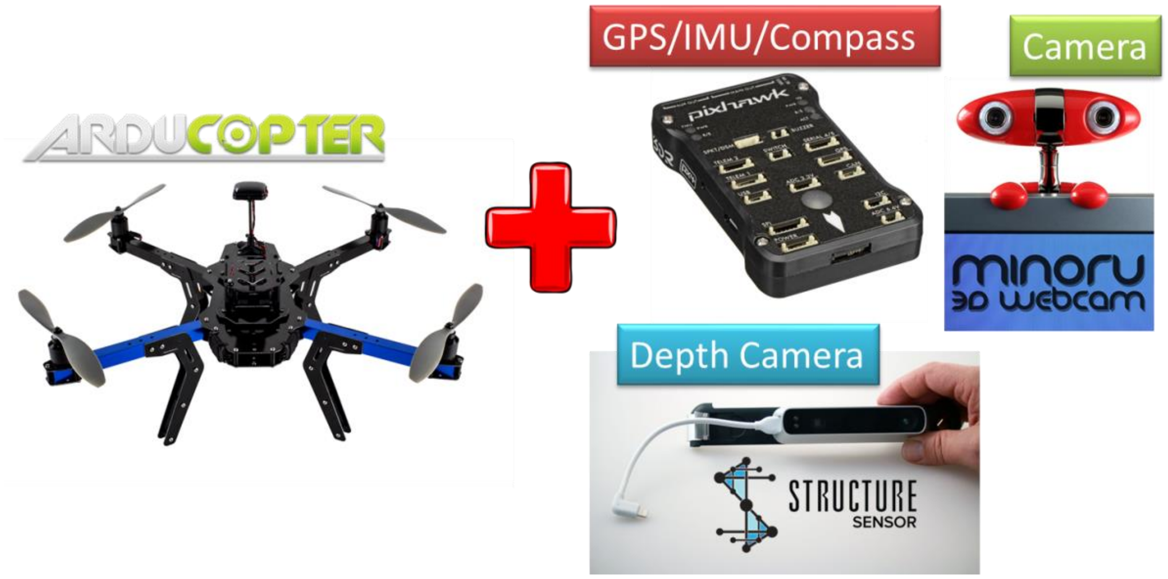


Figure 5.14. Components of the developed indoor/outdoor UAV.

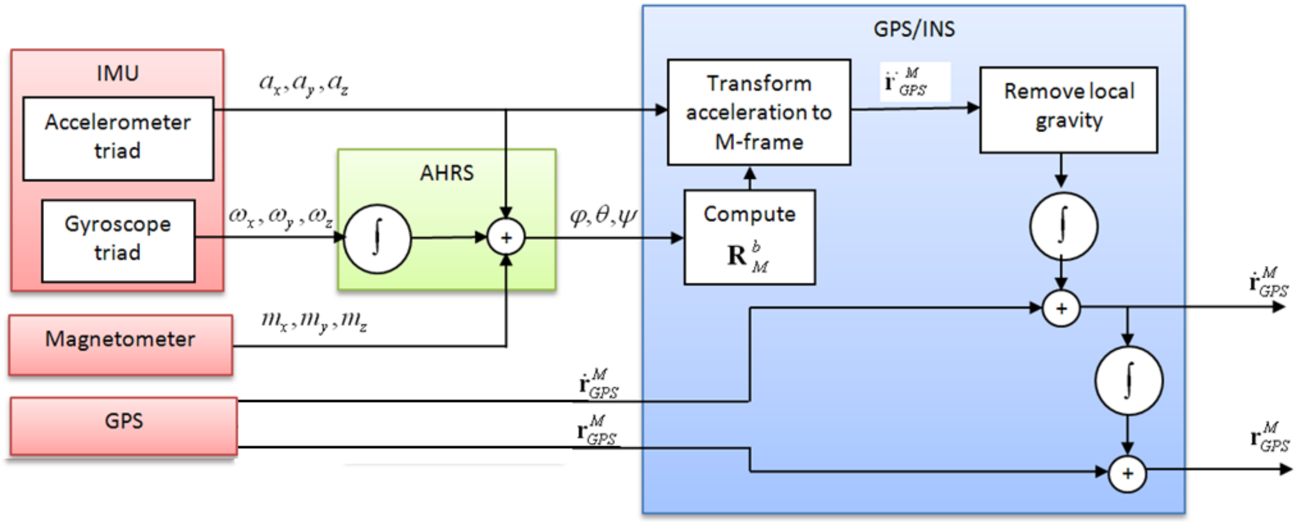


Figure 5.15: Sensor integration strategy.

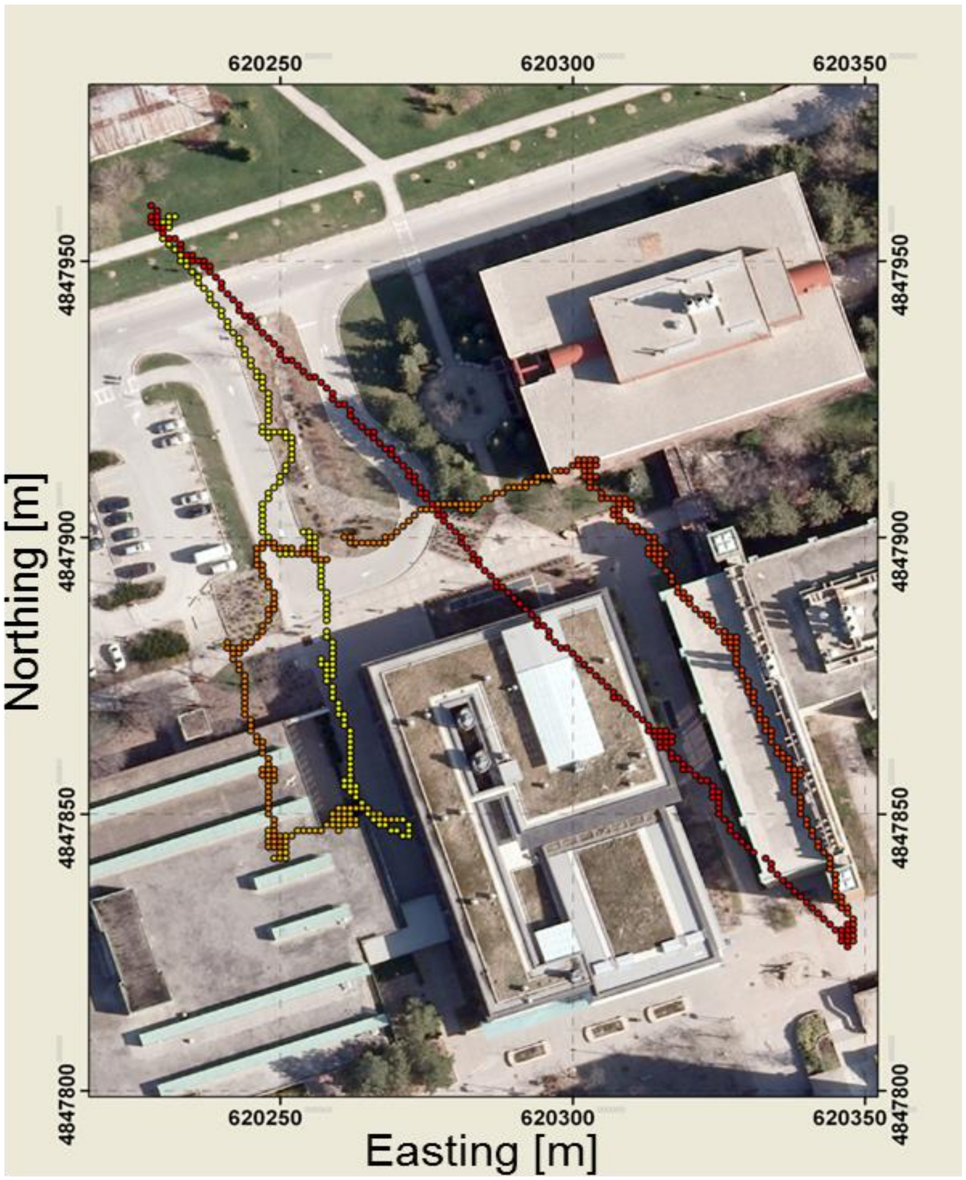


Figure 5.16. The UAV's flight path over the Lassonde Building.

The camera's intrinsic parameters were calibrated beforehand and held fixed in the pose estimation process. The image frames were tested at 480p, 720p, and 1080p resolutions. The trials revealed that, at 40 metres above ground level, 480p video performed the best in terms of processing speed and tracking performance. The tracking performance improved because the noisy edges detected by the moving edges tracker at higher resolutions were removed at lower resolutions, leaving only the stronger edge responses.

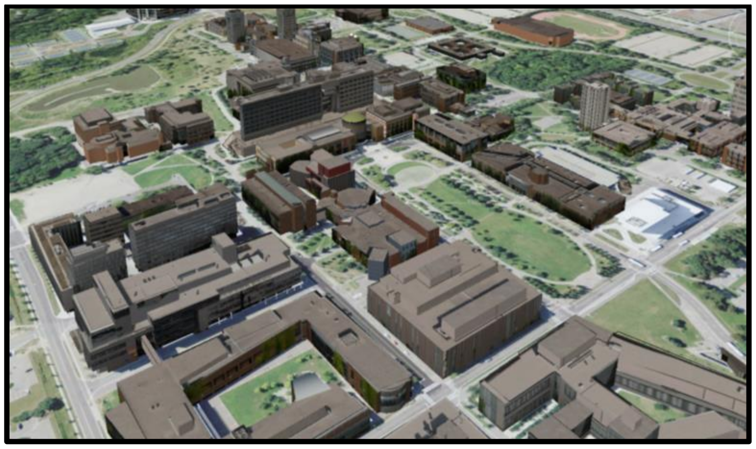


Figure 5.17. York University's 3D campus model.

Figure 5.18 A) to F) shows several results from the tracking along the West side and North side of the Lassonde Building. The red lines are the projections of the 3D building model onto the image plane using the respective camera pose. It is evident that, although the tracker does not diverge, there is misalignment between the image and the projected model; this is due to error in the camera pose. Figure 5.19 A) to F) shows results from the ViSP along the East side of the Lassonde Building. Notably, from frame E) to F) in both Figure 5.18 and Figure 5.19, ViSP's robust control law decreases the misalignment between the image and model. However, these same frames reveal that ViSP is unable to handle occlusions. That is, parts of the 3D model that cannot be seen in the image are being projected onto the image. This causes errors in the camera pose when ViSP tries to match these lines with lines in the image.

Figure 5.20 shows a comparison between the UAV's trajectory according to the onboard GPS versus ViSP's solution. 1137 frames were processed at 6 frames per second. Figure 5.21 A) C) shows the positional coordinates in the X, Y, and Z axes, respectively, from the GPS solution and ViSP's solution. Figure 5.21D) shows the coordinate differences between the two solutions, and Table 5.1 provides the average coordinate difference, along with the standard deviation. The average 3D difference between GPS and VISP is 11.07 m ±15.51 m. With the L1 GPS noise around 3 metres, these errors are high considering the geometric accuracy of the model is 10 to 40 cm. Figure 5.21 shows gaps in ViSP's trajectory from when tracking was lost as a result of a lack of model features in the camera's field of view, and because of rapid motions due to the unstabilized camera.

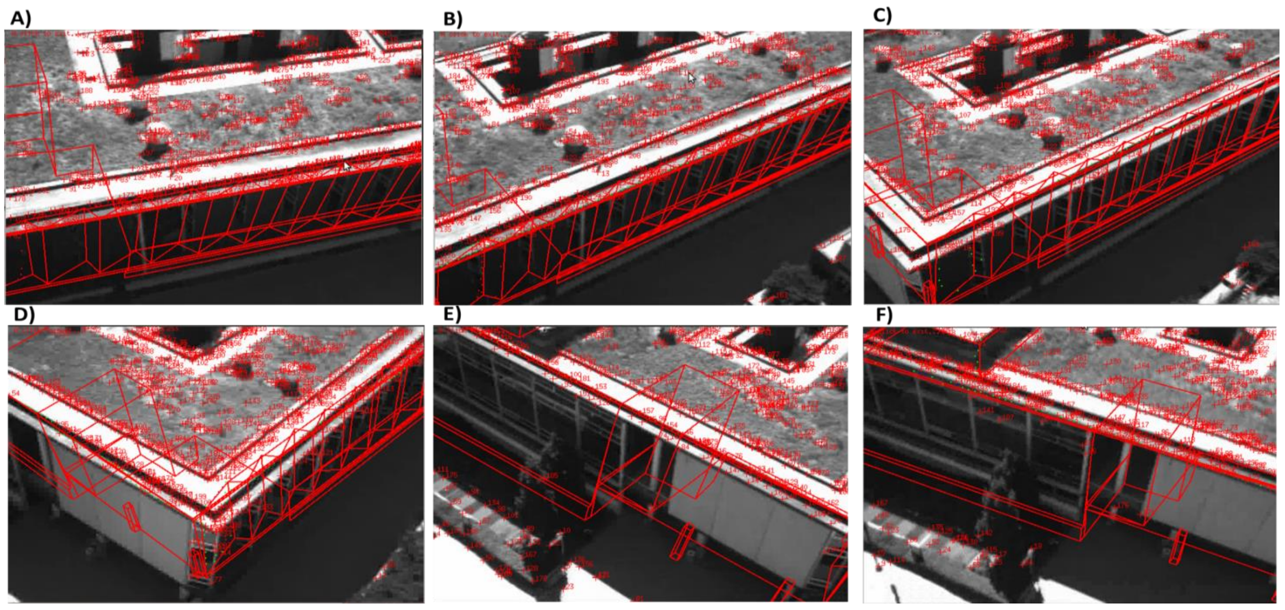


Figure 5.18. Sample frames demonstrating ViSP's model-based tracker. The West side (A-C) and North side (D-F) of York University's Lassonde Building are being observed, the 3D building model is projected onto the image plane (red lines) using the respective camera intrinsic and extrinsic parameters. Harris corners (red crosses with IDs) are also being tracked.

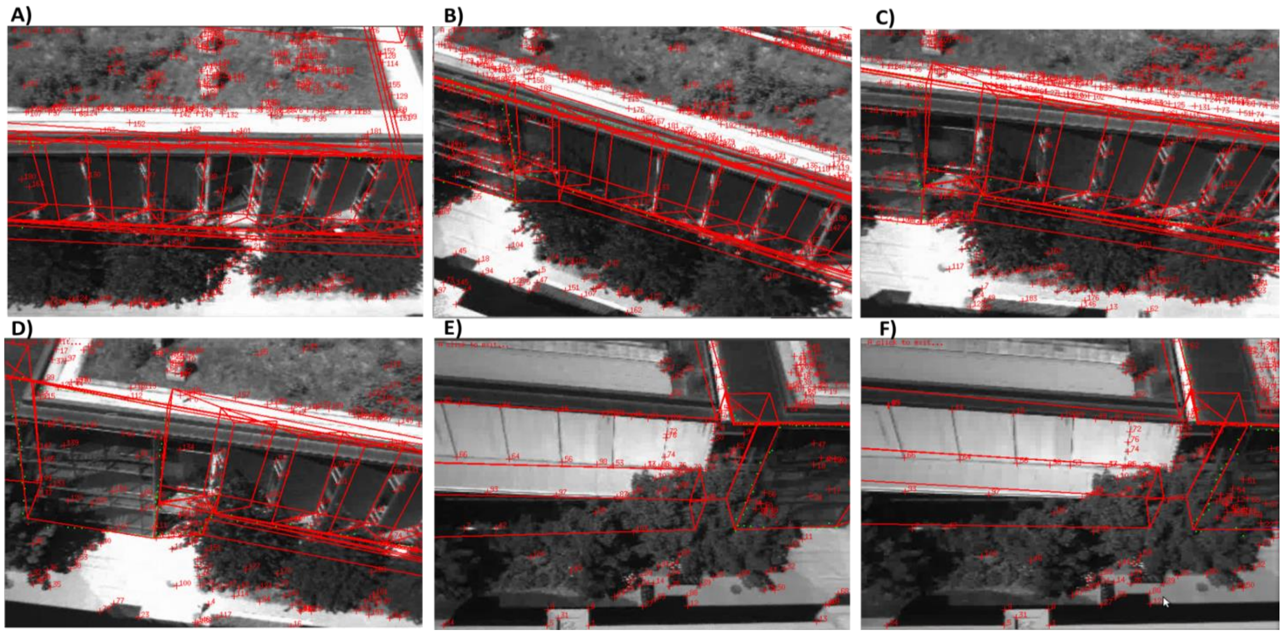


Figure 5.19. Additional sample frames demonstrating ViSP's model-based tracker. The East side of York University's Lassonde Building is being observed, the 3D building model is projected onto the image plane (red lines) using the respective camera intrinsic and extrinsic parameters. Harris corners (red crosses with IDs) are also being tracked.

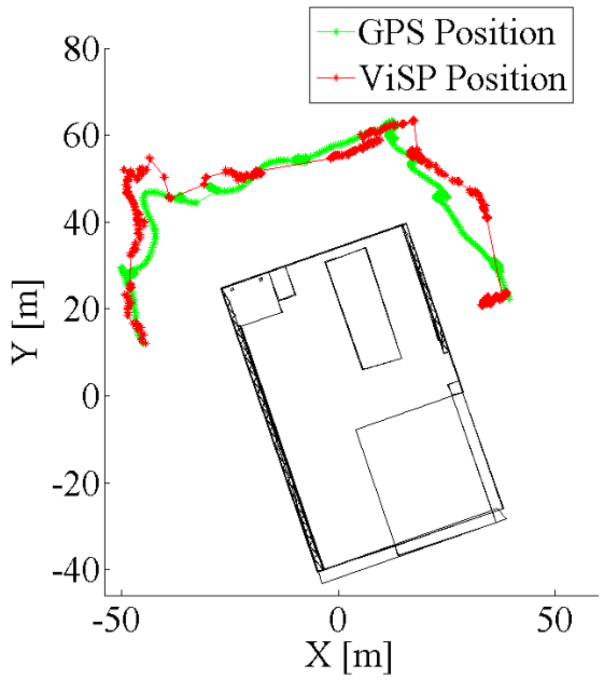


Figure 5.20. An overhead view of the UAV's trajectory according to the onboard GPS (green) and ViSP solution (red).

Table 5.1. Mean and standard deviation for the difference in the UAV's positional coordinates from its onboard GPS and ViSP.

	Mean	**Standard Deviation**
ΔX [m]	8.78	±9.23
ΔY [m]	-1.19	±12.16
ΔZ [m]	6.65	±2.73
Δ3D [m]	11.07	±15.51
$\Delta\psi[^{o}]$	0.15	±13.87

The camera was attached the UAV using a pan-tilt mount, so the camera pitched and yawed independently from the UAV. Therefore, the AHRS' roll and pitch axes did not align with the camera frame. However, both the AHRS' and ViSP's the direction cosine matrix rotation sequences are ZYX, thus their Z-axes are parallel. The camera did not pan throughout the flight, so there was a constant offset between the AHRS and camera yaw angles (60.125^{o}). Figure 5.22A) shows the yaw angles, after removing the offset, and Figure 5.22B) shows the difference. It is evident that ViSP's yaw estimate agrees well with the AHRS solution. The jumps in the data again show where tracking was lost and reset. Table 5.1 provides the mean difference in yaw ($\Delta\psi$), along with its standard deviation.

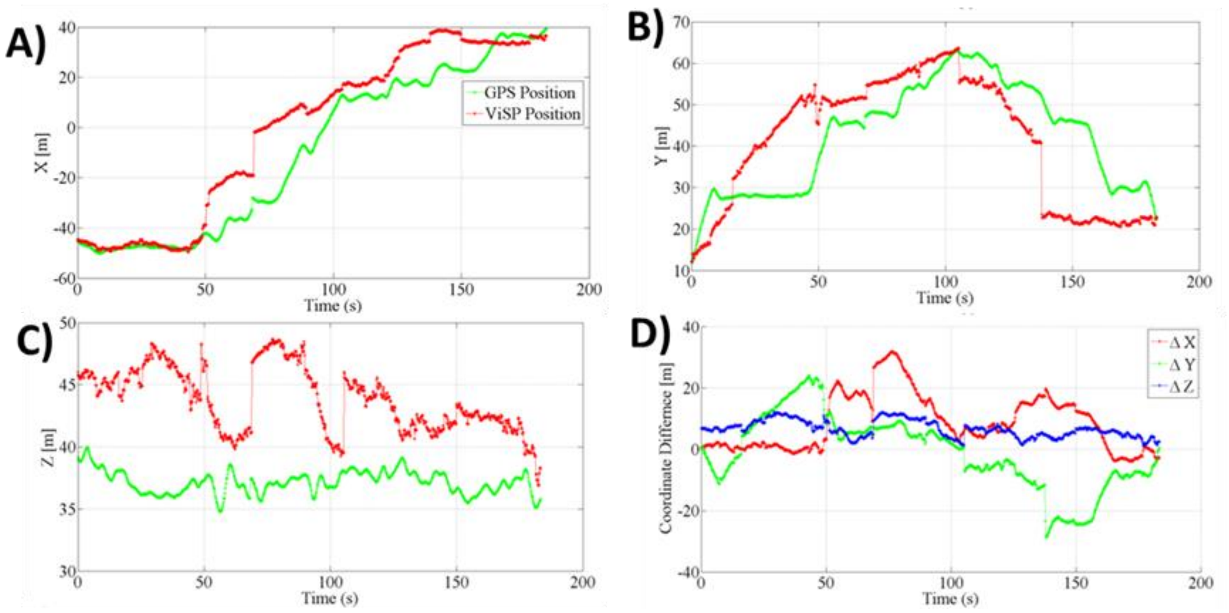


Figure 5.21. Positional comparison between the GPS solution and ViSP's solution.

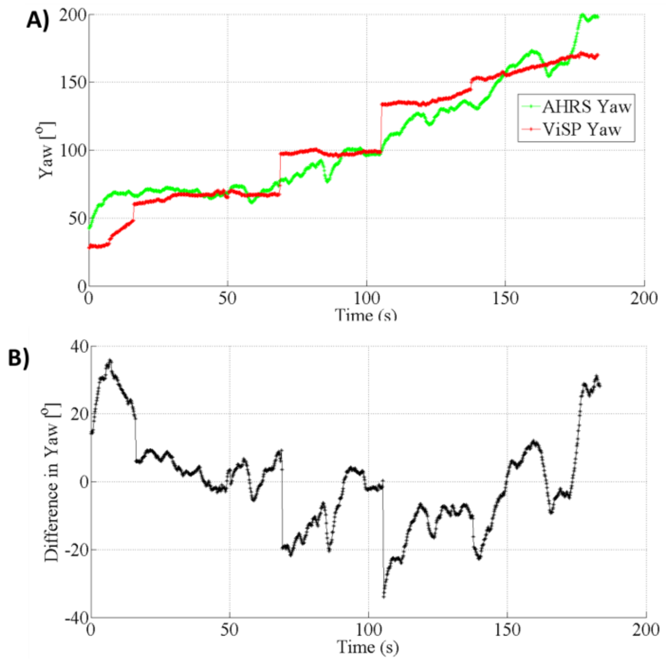


Figure 5.22. Yaw angle comparison between the GPS solution and ViSP's solution.

5.6 Results: Combining ViSP and RGB-D SLAM

The previous experiment yielded larger than expected pose standard deviations, and tracking was often lost throughout the image sequence. The following experiments demonstrate that the localization system's performance is improved, in terms of accuracy and tracking maintenance, by combining incremental map-based (ViSP) with map-building (RGB-D SLAM) localization.

Two experiments are presented in this section. The first experiment demonstrates the ability of the combined ViSP / RGB-D SLAM system to maintain ViSP's tracking when the 3D wireframe model is not in the camera's field of view. The second experiment is an attempt to demonstrate that the combined ViSP / RGB-D SLAM system is capable of accurately mapping large areas without RGB-D SLAM performing loop closures. As previously discussed in Section 5.4 this provides two benefits: Firstly, by closing loops on the 3D wireframe model instead of the RGB-D SLAM map, the amount of data and the amount of time spent collecting data is reduced because there is no need to re-observe the same areas multiple times to generate large loop closures. Secondly, and consequently, without the need for RGB-D SLAM's loop closures, its computationally expensive frame-to-frame comparisons to random frames, and to graph neighbour frames can be set to zero, thus turning RGB-D SLAM into a computationally efficient visual odometry system. Notably, searches for small loop closures between the current frame and its previous 10 frames were performed. This has been shown to reduce the drift in the pose estimate without much computational cost (Endres et al., 2014).

Figure 5.23 shows the strategy for integrating ViSP and RGB-D SLAM. Firstly, the user initializes ViSP by manually selecting at least 4 points in the first image frame that correspond to pre-specified points in the 3D wireframe model. ViSP's VVS algorithm uses these image measurements to estimate the camera pose in the 3D wireframe model's coordinate system. RGB-D SLAM then begins mapping and tracking the camera pose, and continues until ViSP's ME algorithm detects corresponding image and model feature points. ViSP's VVS algorithm then corrects the camera pose and RGB-D SLAM is reset at ViSP's provided pose. Upon resetting, RGB-D SLAM saves each depth frame, with its corresponding camera pose, that was collected since the last reset.

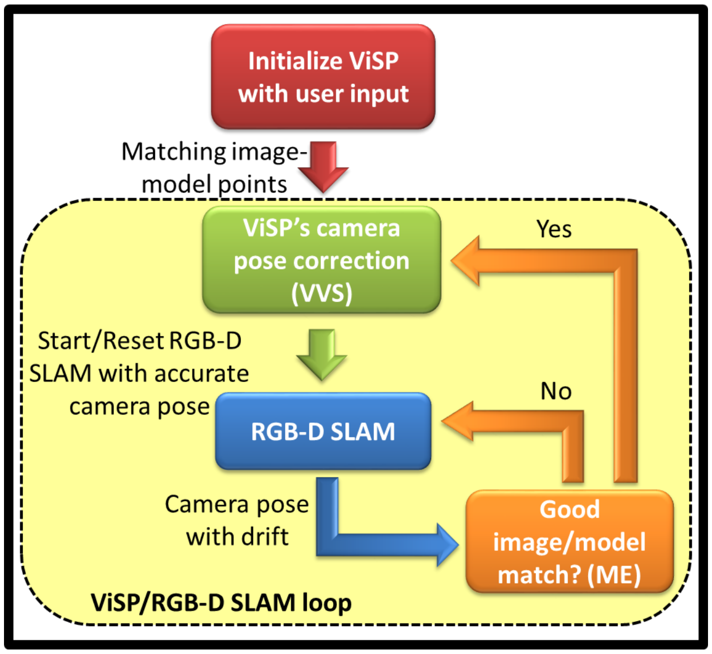

Initialize ViSP with user input
Matching image-model points
ViSP's camera pose correction (VVS)
Yes
Start/Reset RGB-D SLAM with accurate camera pose
No
RGB-D SLAM
Camera pose with drift
Good image/model match? (ME)
ViSP/RGB-D SLAM loop

Figure 5.23. ViSP / RGB-D SLAM integration workflow.

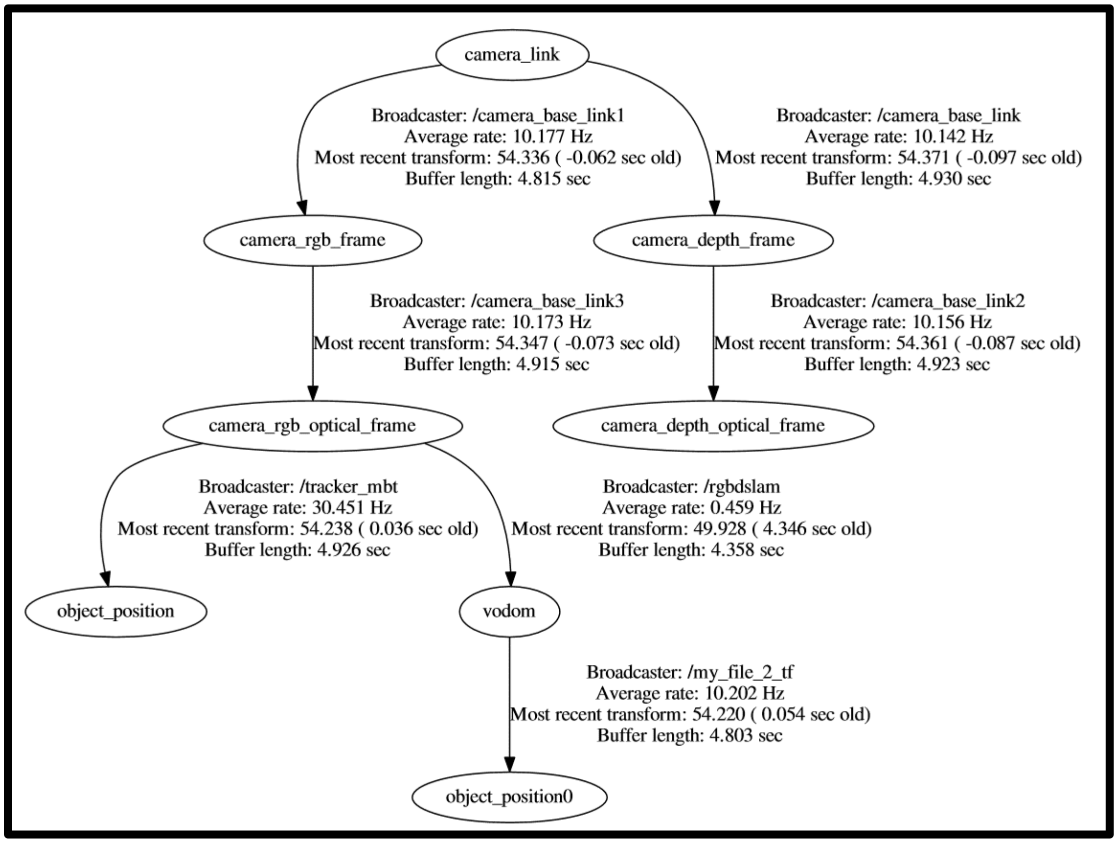

camera_link
Broadcaster: /camera_base_link1
Average rate: 10.177 Hz
Most recent transform: 54.336 (-0.062 sec old)
Buffer length: 4.815 sec
Broadcaster: /camera_base_link
Average rate: 10.142 Hz
Most recent transform: 54.371 (-0.097 sec old)
Buffer length: 4.930 sec
camera_rgb_frame
camera_depth_frame
Broadcaster: /camera_base_link3
Average rate: 10.173 Hz
Most recent transform: 54.347 (-0.073 sec old)
Buffer length: 4.915 sec
Broadcaster: /camera_base_link2
Average rate: 10.156 Hz
Most recent transform: 54.361 (-0.087 sec old)
Buffer length: 4.923 sec
camera_rgb_optical_frame
camera_depth_optical_frame
Broadcaster: /tracker_mbt
Average rate: 30.451 Hz
Most recent transform: 54.238 (0.036 sec old)
Buffer length: 4.926 sec
Broadcaster: /rgbdslam
Average rate: 0.459 Hz
Most recent transform: 49.928 (4.346 sec old)
Buffer length: 4.358 sec
object_position
vodom
Broadcaster: /my_file_2_tf
Average rate: 10.202 Hz
Most recent transform: 54.220 (0.054 sec old)
Buffer length: 4.803 sec
object_position0

Figure 5.24. ViSP / RGB-D SLAM tree of coordinate transforms output from ROS's *tf/view_frames*.

The Robot Operating System (ROS) was used to run and integrate the Freenect Kinect driver, RGB-D SLAM, and ViSP. The *visp_tracker* package's ROS topic *object_position_hint* was used to supply ViSP with RGB-D SLAM's camera pose estimates. This topic allowed ViSP to maintain tracking when it was not able to estimate its own camera pose. A custom ROS package was developed to provide RGB-D SLAM with the camera poses estimated by ViSP, allowing RGB-D SLAM to start/reset in the 3D wireframe model's coordinate system. The ROS transform (*tf*) tree is shown in Figure 5.24, where *my_file_2_tf* generates the transformation from the 3D wireframe model's coordinate system (object_position0) to RGB-D SLAM's coordinate system (vodom).

5.6.1 Experiment 1: Improving tracking maintenance

The first experiment was performed in the Bergeron Entrepreneurs in Science and Technology (BEST) lab. The 3D wireframe model consisted of the rectangular outline of the door, shown in green on the right of Figure 5.25. The geometric accuracy of the model was 2 to 5 centimetres. For testing purposes, a Microsoft Kinect Sensor for Windows 1517 was used to collect the data. The Kinect comprises of an RGB camera producing images of 640x480 pixels at 30 frames per second, and an infrared (IR) emitter and an IR depth sensor, which use a structured light approach to output depth images of 640x480 pixels at 30 frames per second.

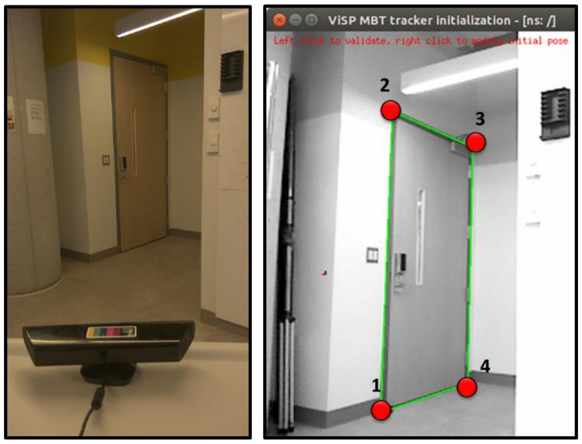

ViSP MBT tracker initialization - [ns: /]
Left click to validate, right click to modify initial pose
2
3
1
4

Figure 5.25. Initializing ViSP in the first experiment using the four corners of the door to the BEST lab. The 3D wireframe model of the door is projected onto the image (green lines) using the estimated camera pose.

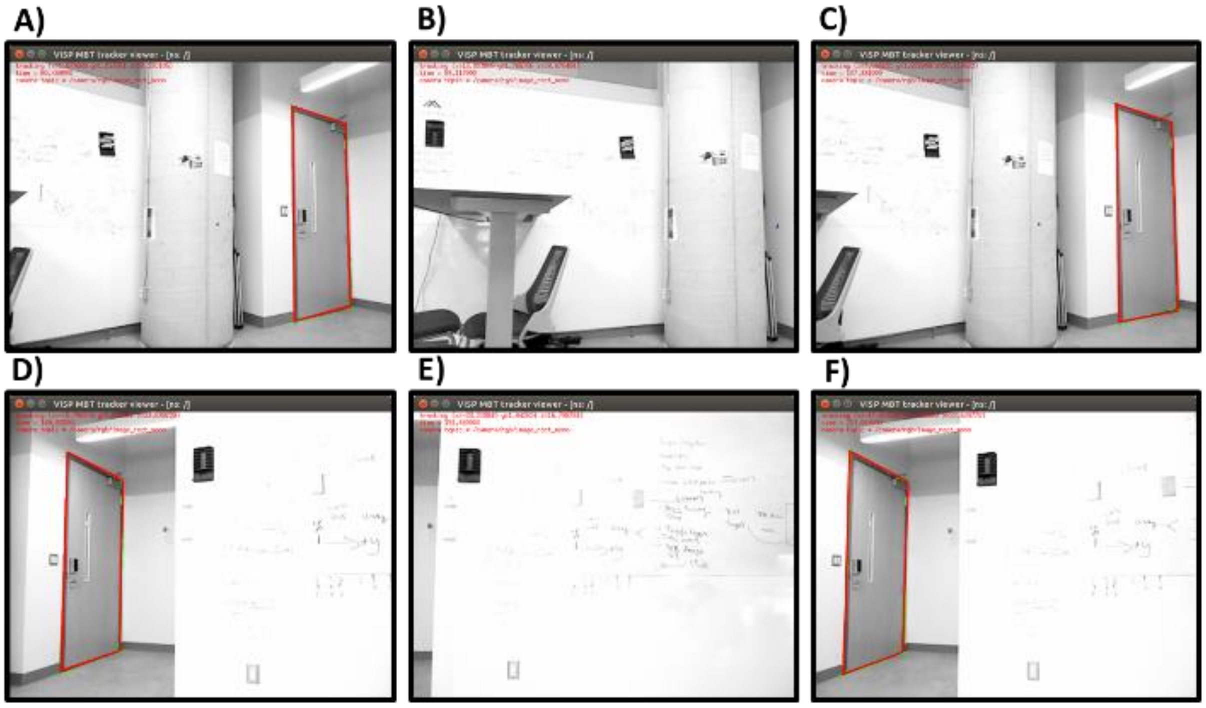


Figure 5.26. Sample frames from the first experiment demonstrating ViSP's model-based tracker aided by RGB-D SLAM. The door of York University's BEST lab leaves the field of view in B) and E). The door re-enters the field of view in Frames C) and F), showing that tracking the pose was not lost.

Figure 5.26 shows the tracking in progress, where the model of the door (red lines) is visible in A). In B), the model leaves the field of view as the camera rotates towards the left, but because RGBD-SLAM is concurrently providing pose estimates to VISP, ViSP does not fail as it would while running alone. In C), the model reappears into the field of view as the camera rotates towards the right, showing that tracking the pose was never lost. D) to F) repeat the demonstration, but first moving right, then left. The pose estimates retrieved from ViSP were used to align the RGB-D SLAM point cloud with the model in real-time (Figure 5.27A). CloudCompare was used to measure the spatial distance between the model and point cloud (Figure 5.27C). Focusing on the door where ViSP performs the drift correction, 3 cm mapping accuracy was achieved (Figure 5.27D). This was an acceptable results considering the geometric accuracy of the 3D model is 2 to 5 cm.

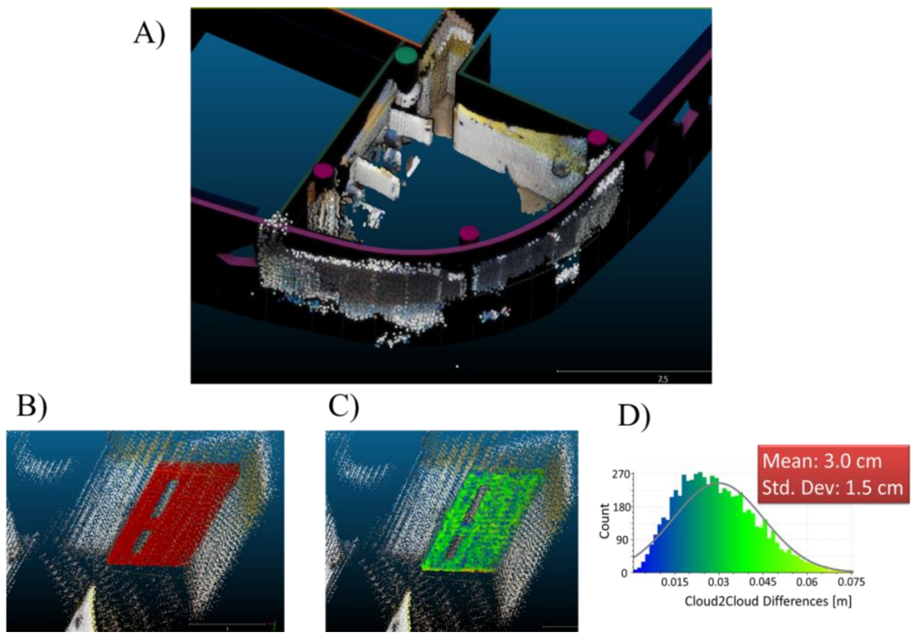


Figure 5.27. A) RGB-D SLAM point cloud aligned with the 3D model using the pose estimates from ViSP. B) 3D model of a door (shown in red) used to generate the simplified rectangular model used in the first experiment. C) CloudCompare's cloud-to-cloud (C2C) difference measurement between the 3D model of the door and the RGB-D SLAM point cloud. D) Histogram of C2C differences, with mean and standard deviation.

5.6.2 Experiment 2: Tracking pose across large areas

The second experiment was performed on the second floor of York University's Bergeron Centre for Engineering Excellence. The 3D CAD design plan of the Bergeron Centre of Engineering Excellence was used as the known 3D map of the environment. It is a TIN model consisting of the building's architectural components (walls, windows, doors, etc.), and structural components (concrete slabs and pillars, etc.). The geometric accuracy of the model was 2 to 5 centimetres.

Figure 5.28 shows the 3D wireframe model that was used in this experiment. This model was extracted from the full 3D model of the Bergeron Centre. ViSP was initialized using the four corners of the red door. The Kinect then travelled down the hallway towards the green door, and finally to the blue door. Notably, the green door is the entrance to the BEST lab, the door used in the first experiment. Figure 5.29 shows 6 sample frames that were captured during the traverse. The red door is visible in A) and B). The green door appears in B) to D). The blue door can be seen in E) and F).

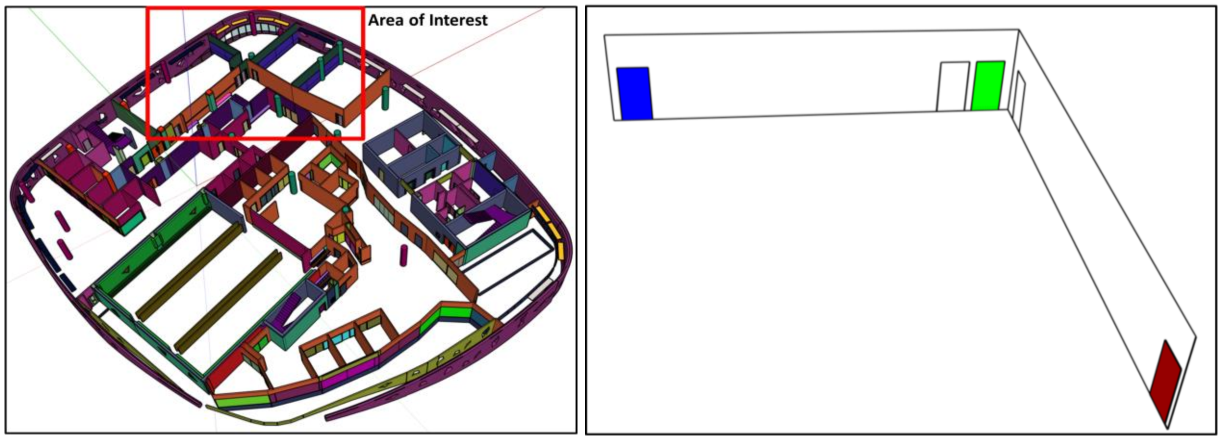


Figure 5.28. Left) The 3D indoor model of the second floor in the Bergeron Centre. The area of interest of the second experiment is outlined by the red box. Right) The 3D wireframe model used in the second experiment.

This experiment revealed that increases in the frequency of ViSP camera pose corrections resulted in trajectories with less drift, and increased overall processing speeds. Conversely, the computer slowed down when RGB-D SLAM ran for extended periods of time without resetting. This was due to the large amounts of data being processed simultaneously. Notably, Figure 5.29B) shows a slight misalignment between the image and the projected 3D model. This is due to drift in the pose generated by RGB-D SLAM. However, RGB-D SLAM often corrected this error by closing small loops in every 10 frames.

The main disadvantage of ViSP is evident in Figure 5.29 C) and E): The portion of the 3D model that is behind the camera is inverted and projected onto the image plane, causing large errors in pose estimation when the ME tracker matches these edges with edges in the image. Although ViSP is designed to recognize these point as outliers, it is not foolproof. Thus, future work may involve supressing ViSP from projecting parts of the model that are behind the camera.

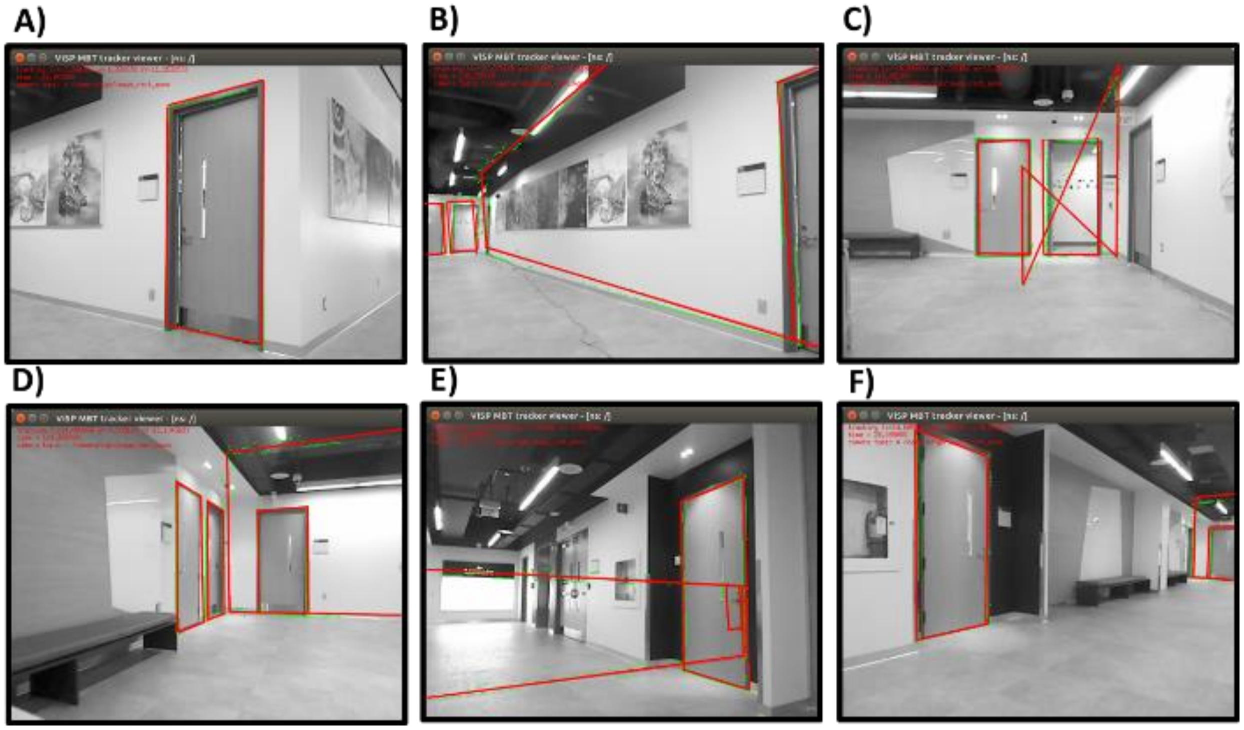


Figure 5.29. Sample frames from the second experiment demonstrating ViSP's model-based tracker aided by RGB-D SLAM. B) shows a misalignment between the image and the projected 3D model. This is due to error in the pose generated by RGB-D SLAM. However, RGB-D SLAM often corrected this error by closing small loops in every 10 frames. C) and E) show the portion of the 3D model that is behind the camera is inverted and projected onto the image plane. Tracking was not lost as ViSP is designed to recognize these points as outliers.

5.7 Concluding remarks

The performance of ViSP was assessed in tracking a building model in an outdoor environment from the camera of a UAV. Throughout the image sequence, ViSP often lost tracking as a result of a lack of model features in the camera's field of view, and because of rapid motions due to the unstabilized camera. Further, the average 3D difference between GPS and VISP solutions was 11.07 m ±15.51 m. With the L1 GPS noise around 3 metres, these errors are high considering the geometric accuracy of the model is 10 to 40 cm. The presented sample frames showed that ViSP was unable to handle occlusions. That is, parts of the 3D model that cannot be seen in the image are being projected onto the image. Poor occlusion handling caused large errors in the camera pose when ViSP tried to match these lines with lines in the image. A maintained misalignment between the image and the model throughout the image sequence also added a bias in the camera pose. The work in this section has contributed to (Li-Chee-Ming and Armenakis, 2015).

The experiments suggested that ViSP can provide only an initial approximation for camera pose because the moving edges tracker is a local matching algorithm. A simultaneous bundle adjustment will globally optimize the camera poses, leading to an increase in the accuracies. Figure 5.18 and Figure 5.19 show red points that are labelled with their ID numbers, these are Harris corners being tracked using a KLT tracker (Lucas and Kanade, 1981). They may be used as tie points in the bundle adjustment to further optimize the camera poses and generate a sparse point cloud. Dense matching (Rothermel et al., 2012) could be used to increase the point cloud's density. The Iterative Closest Point (ICP) algorithm (Besl and McKay, 1992) may then be used to refine the alignment between the point cloud and the 3D building model; this should further increase the camera pose accuracies. As large-scale bundle adjustment, dense matching, and ICP are computationally intensive, they should be done in a post-processing stage (further explained in Section 9.1).

The real-time camera pose accuracies can be improved by augmenting ViSP's pose estimate with another pose estimation system. For instance, one could integrate ViSP's pose with the positional and orientation data from the UAV's onboard GPS and AHRS, respectively, through a Kalman filter, to increase the accuracy of the pose and bridge GPS gaps.

It was then demonstrated that by integrating ViSP with RGB-D SLAM, not only does RGB-D SLAM improve ViSP's tracking performance, but ViSP substitutes RGB-D SLAM's loop closing, while improving runtime performance. Specifically, 3 cm mapping accuracies were achieved in an indoor environment and ViSP did not fail when there were no model features in the image as it would while running alone. Further, ViSP's pose estimation process can be thought of as loop-closing on the 3D model. Consequently, RGB-D SLAM's loop closing parameters may be set to low values, thus gaining computational efficiency without sacrificing localization accuracy. Unfortunately, an implementation issue was encountered when using a larger 3D model. That is, parts of the 3D model that were behind the camera were inverted and projected onto the image plane, causing large errors in the pose when the tracker matched these model lines with lines in the image. As a result of this behaviour, reasonable pose estimates could not be obtained. In conclusion, ViSP cannot be used when the model surrounds the camera. The work in this section has contributed to (Li-Chee-Ming and Armenakis, 2016).

The limitations of ViSP revealed in the presented experiments led to the second contribution of this dissertation: A map-based incremental localization algorithm, which aimed to building upon ViSP’s weaknesses.

6 A MAP-BASED INCREMENTAL LOCALIZATION ALGORITHM

The previous chapter assessed the performance of state-of-the-art model-based trackers in larger environments, tracking larger and more complex objects. Specifically, an open source software implementation of a state-of-the-art edge-based model tracker, called ViSP (Visual Servoing Package) (Marchand and Chaumette, 2002), was used to track the outdoor facades of a 3D building model using the RGB camera of a UAV. Section 5.5 showed that ViSP was also used to track a 3D indoor model using the RGB camera of a UAV. In both situations, the tracker generated the camera's pose for each image frame with respect to the 3D model's georeferenced coordinate system. The results revealed ViSP's shortcomings, which explained why the sizes of objects and working environments in previous work were limited in size and complexity. The experiments provided in Section 5.5 revealed gaps in the trajectory where tracking was lost. Lost tracking was due to a lack of model features in the camera's field of view, and also because of rapid camera motion. Further, the pose estimate was biased due to incorrect edge matches.

This chapter presents the second contribution of this dissertation: An improved model-based tracker that overcomes the aforementioned deficiencies of the state-of-the-art. The experiments demonstrate two novel applications of the proposed model-based tracker:

1) An outdoor localization solution for bridging GNSS gaps and aiding the navigation sensors (GNSS/INS) using only one RGB camera.
2) An indoor localization solution for UAVs using only one RGB camera.

The proposed map-based incremental localization system uses the Gazebo Robot Platform Simulator (Open Source Robotics Foundation, 2017) for occlusion handling. Gazebo is a 3D simulator with a physics engine, capable of simulating robots and a variety of sensors in complex and realistic indoor and outdoor environments. It is typically used to test algorithms and design robots (Figure 6.1). An additional contribution of this dissertation is utilizing Gazebo in real-time applications.

As the model-based tracker operates, the pose of a simulated camera is dynamically updated in the Gazebo simulator using the previously estimated pose (Figure 6.2). The proposed model-based tracker automatically matches features from the real camera image with features in the simulated camera image, referred to herein as a synthetic image, to estimate the current pose. The following section describes the workflow of the proposed tracker in more detail.

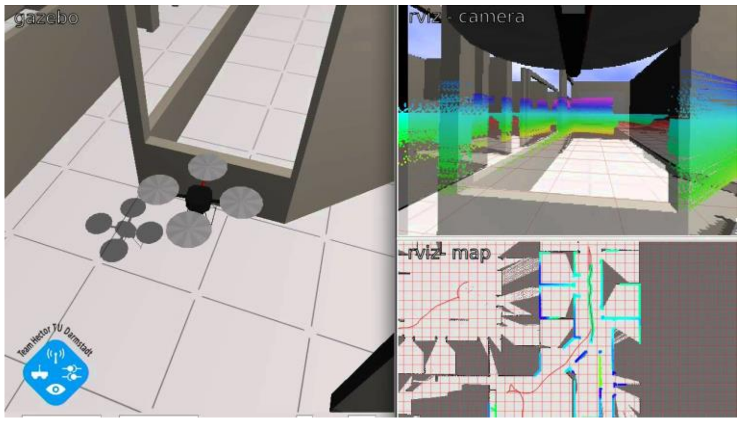


Figure 6.1. Simulated quadcopter in Gazebo's virtual environment (Open Source Robotics Foundation, 2017).

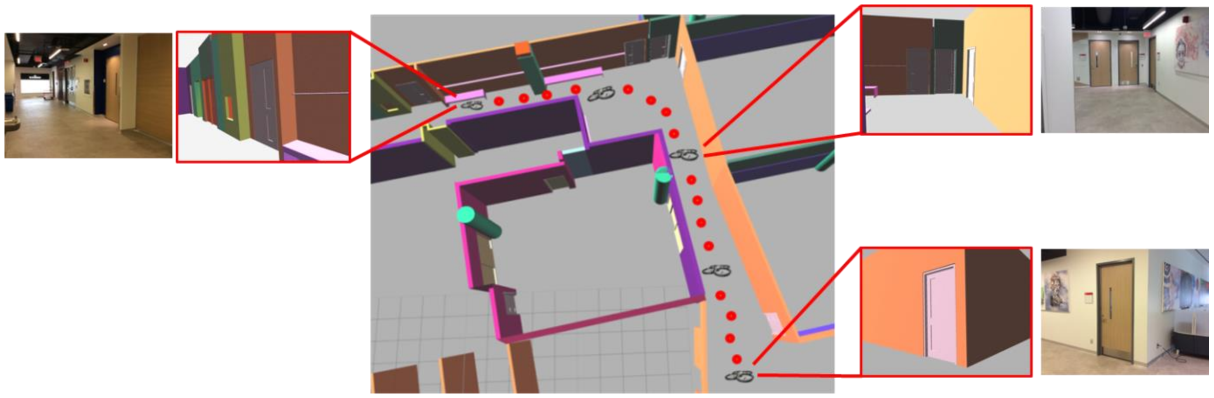


Figure 6.2. The trajectory of the UAV within the 3D CAD model of the building.

6.1 Pose Estimation Workflow

This section presents a visual localization algorithm that automatically identifies corresponding features between the images streamed from a UAV flying in an urban environment and a 3D model of the environment. The camera pose resolved from the corresponding feature points provides a precise trajectory estimate with respect to the 3D environment's coordinate system. The following section, accompanied by Figure 6.3, describes the pose estimation process.

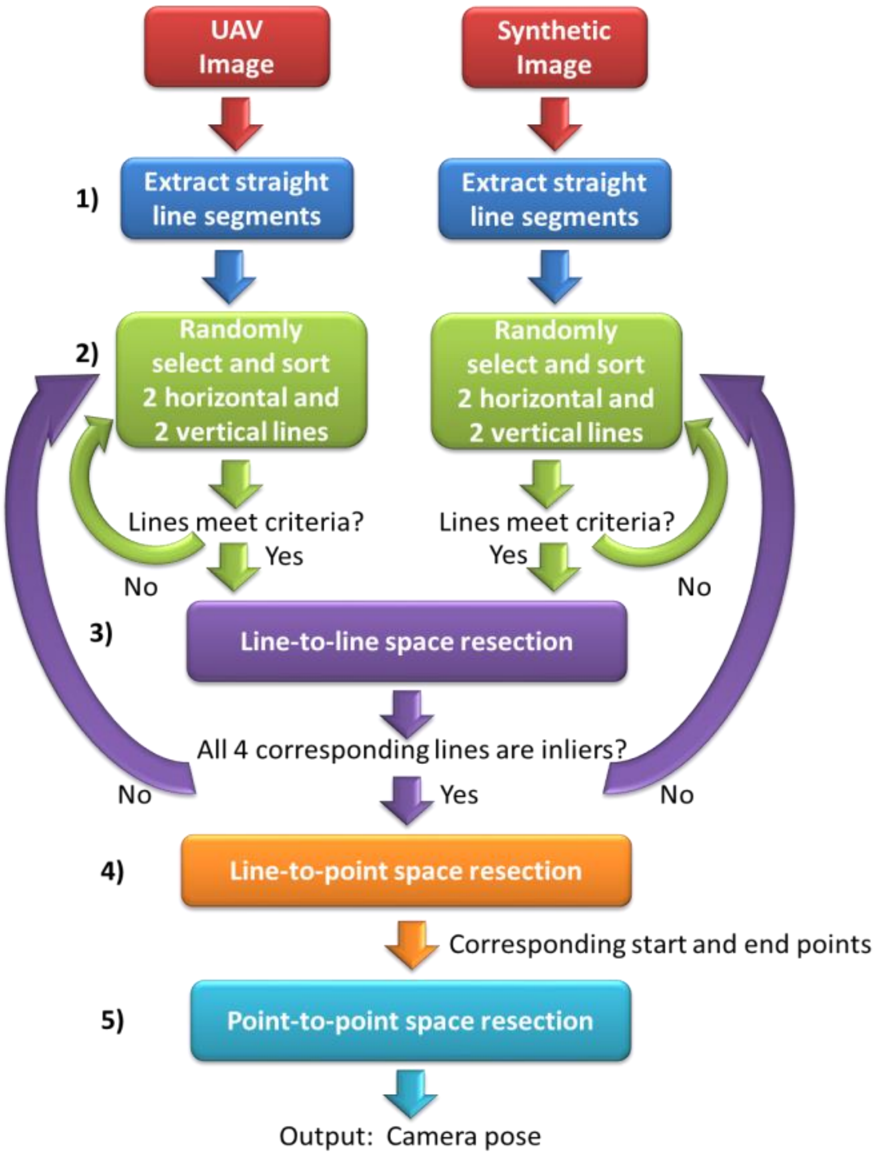

UAV Image
Synthetic Image
1)
Extract straight line segments
Extract straight line segments
2)
Randomly select and sort 2 horizontal and 2 vertical lines
Randomly select and sort 2 horizontal and 2 vertical lines
Lines meet criteria?
Lines meet criteria?
Yes
Yes
No
No
3)
Line-to-line space resection
All 4 corresponding lines are inliers?
No
Yes
No
4)
Line-to-point space resection
Corresponding start and end points
5)
Point-to-point space resection
Output: Camera pose

Figure 6.3. The proposed incremental map-based localization algorithm.

1) The user must provide an initial approximation of the camera pose for the first frame. Straight line segments are then extracted from the image. These lines are referred to as real image lines. The very simple and efficient line extraction techniques presented by Smith et al. (2006), such the length, midpoint, quarter point and Bresenham walk checks, are used. This line extraction method was selected because it was designed to function in real-time, and it is capable of

extracting long, uninterrupted lines, and merging lines based on proximity and parallelism. The extracted lines are then matched with lines extracted from a virtual camera's synthetic image using the same line extraction approach; these are referred to as virtual image lines. This virtual camera is positioned in Gazebo's environment using the previous frame's estimated pose. Both real and virtual image lines are represented by their 2D start and end points.

2) In the matching phase, two horizontal lines and two vertical lines are randomly selected from both the real and virtual image lines until the lines satisfy the following conditions: i) all lines must be longer than a user specified length, ii) the vertical separation across the image between the two horizontal lines must be larger than a user specified distance, and iii) similarly, the horizontal separation between the two vertical lines must be larger than a user specified distance. These three conditions aim to yield the minimum number of lines that can be used to accurately and reliably estimate the camera pose. For both the real and virtual lines, the two horizontal lines are sorted top to bottom, and the two vertical lines are sorted left to right. The top real horizontal line is then paired with the top virtual horizontal line. Similarly, the bottom real horizontal line is paired with the bottom virtual horizontal line, the left real vertical lines and the left virtual vertical lines are paired, and finally the right real vertical line is paired with the right virtual vertical line.

3) The correspondence of these four real-virtual line pairs is tested by a line-to-line (L2L) space resection (Meierhold et al., 2008). This method of space resection estimates the camera pose given corresponding 2D image lines and 3D object lines. 3D object lines are referred to in this work as virtual object lines and are represented by their 3D start and end points. In this case, the 3D start and end points of a virtual object line are calculated via ray-casting, that is, by casting a ray starting from the virtual camera, through its respective 2D image point, and determining where it intersects with the 3D model. The real and virtual image lines are considered correctly matched if the pose covariances from the space resection are within a user defined tolerance and if the estimated camera pose is within the error ellipsoid of the camera pose predicted by the Kalman filter of the navigation system. If either condition is not met, four new lines are randomly sampled from the sets of real and virtual lines. Matching lines via this line-to-line resection was chosen because it does not require the corresponding lines to have corresponding

start and end points; a condition which is seldom met, due to occlusions for instance. Section 6.5.2.3 elaborates on this point.

4) Meierhold et al. (2008) demonstrate that a more precise pose estimate can be obtained by a point-to-line (P2L) space resection. This approach is interesting because it not only solves for the camera pose, but also estimates the 3D coordinates of the corresponding object point for each real image line's start and end points, more detail is provided in Section 6.5.2.2.

5) Finally, because corresponding image and object points are now available (from the P2L resection), a traditional point-to-point (P2P) space resection is done. This provides a more precise estimate for the camera pose than the point-to-line approach because the measurement redundancy is higher - the point-to-point resection only estimates the 6 camera pose parameters while the point-to-line space resection estimates the 6 camera pose parameters plus one additional parameter for every 2D image point measurement. As described later in Section 6.5.1, this parameter (t) identifies the point on the 3D model where the ray from the UAV's camera, passing through the real image point, intersects the 3D virtual object line (Meierhold et al., 2008).

For example, Figure 6.4A) shows a sample image captured by the UAV's camera and Figure 6.4B) shows its corresponding synthetic image in the Gazebo simulator. The lines labelled 1 and 2 are two randomly sampled horizontal lines, sorted top to bottom. The lines labelled 3 and 4 are two randomly sampled vertical lines, sorted left to right. Despite not having corresponding start and end points, these four line pairs correspond according to the line-to-line space resection test. Figure 6.4C) shows the image projections of the 3D virtual object lines that were estimated by the point-to-line space resection. It is evident that the start and end points of each real and virtual image line now correspond. These start and end points are then used in a point-to-point space resection to yield the final estimate for this frame's camera pose.

Notably, an assumption is made that vertical lines in the environment remain approximately vertical in the image, implying that horizontal to oblique (not vertical/nadir) imagery is captured from the UAV.

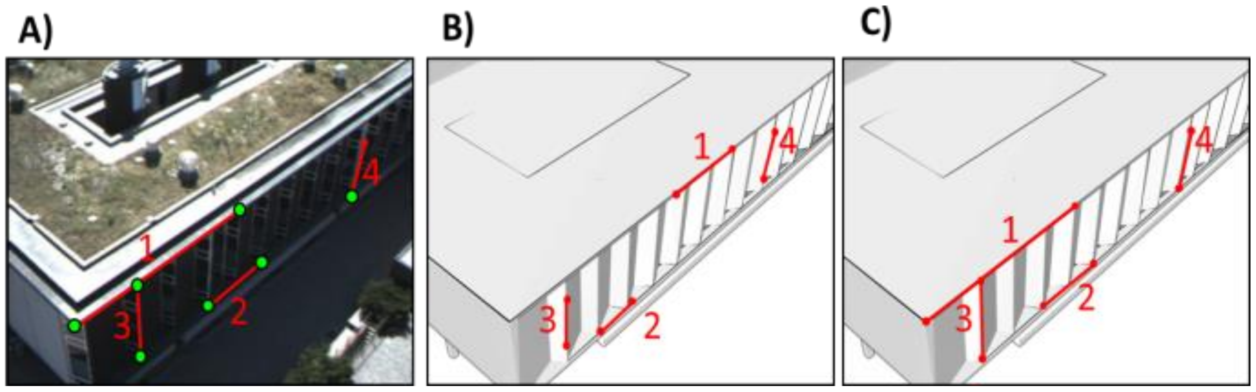


Figure 6.4. Line matching example: Real image (A) and corresponding synthetic image (B), each with 2 randomly sampled horizontal lines, sorted top to bottom (labelled 1 and 2) and 2 randomly sampled vertical lines, sorted left to right (labelled 3 and 4). C) shows the real image lines' corresponding virtual image line start and end points estimated by the line-to-point space resection.

The following sections provide a justification and methodology for extracting lines, and the mathematical descriptions of the pose estimation methods, namely point-to-point space resection (Section 6.5.1), point-to-line space resection (Section 6.5.2.2) and line-to-line space resection (Section 6.5.2.3). Section 6.6 provides experimental results demonstrating an increase in performance, in terms of accuracy and precision, of the proposed incremental map-based localization compared to the state-of-the-art in both indoor and outdoor environments.

6.2 Feature selection

It is evident in Figure 6.5 that the textures of the building models have little correlation with the UAV's imagery because of changes in illumination and weather conditions, along with the presence of shadows. Further, the colours of buildings may change with time, for example, when roofs are repaired or walls are painted. However, the shape of the building remains relatively stable in time and is more suitable for matching. Thus, this work focuses towards matching geometric features (points, lines, polygons, etc.), as opposed to texture-based features such as using SIFT (Lowe, 1999) and SURF (Bay et al., 2008) algorithms. Further, matching geometry is convenient as 3D models generally provide a database of georeferenced vertices (points), edges (lines), and faces (polygons).

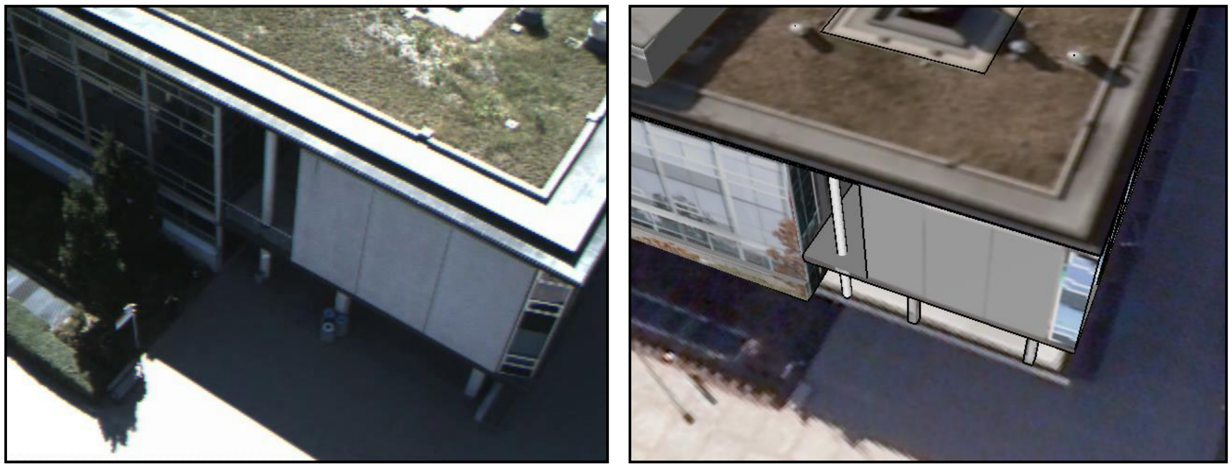


Figure 6.5. Real image (left) and corresponding synthetic image (right).

Extracting feature points with a corner detector, such as Harris corners (Harris and Stephens, 1988), would yield many unwanted points from objects surrounding the building such as trees for instance which are not included in the model. To consider matching polygons, a robust solution is required that could handle a variety of noise, occlusions, and incomplete or unclosed figures. Such a method would increase the complexity and run-time of the system.

Matching silhouettes has been proposed (Latecki et al., 2000), but are limited as they ignore internal contours and are difficult to extract from real images. More promising approaches are the matching of edges based, for example, on the Hausdorff distance (Huttenlocher et al. (1993), Valtkamp and Hagedoorn (1999)) or the Distance Transform (Gavrilla and Philomin, 1999). A drawback for this application is that the method does not return congruent features between the image and the model (Belongie et al., 2002).

Linear features were chosen to establish correspondence between the model and the image. Line matching approaches are divided into algorithms that match individual line segments and algorithms that match groups of line segments. Matching single lines is based on geometric properties, such as orientation, length, and extent of overlap, and thus lines contain richer semantic information than point features. Matching groups of lines takes away the ambiguity involved by providing both pattern associations and geometric information. Graph-matching is a common approach, as graphs capture relationships such as left of and right of, and topological connectedness (Baillard et al., 1999). Various approaches may be used to extract lines from images. Leung et al. (2008) use vanishing point analysis to extract horizontal and vertical lines from an image; however, they demonstrate that extracting vanishing points requires a lot of processing power to realize a real-time system. Other line extraction techniques may also be used, such as the Hough Line Transform (Duda and Hart, 1972), the Progressive Probabilistic Hough Transform (Matas et al., 2000), or the Line Segment Detector (Von Gioi et al., 2010).

6.3 Extracting lines

The chosen linear feature, referred to in this work as the *Vertical Line Feature*, consists of two corners connected by a vertical line. In agreement with Cham et al. (2010), Vertical Line Features were chosen over other features because they commonly appears in structured environments, and the gyro-stabilized camera enables vertical lines to be identified efficiently and reliably, as vertical lines in the object space are highly probable to be vertical or close to in the image space.

The following process was developed to extract Vertical Line Features from images: Firstly, corners are extracted from the Canny edge image and classified according to their orientation into one of the following types: 1) Top Left, 2) Bottom Left, 3) Top Right, or 4) Bottom Right. For example, the Top Left Corner Templates in Figure 6.6A) are used to classify top left corners via 2D cross-correlation (Lewis, 1995). Rotated versions of these templates are used to classify the remaining corner types in a similar fashion, i.e., the Top Right Corner Templates are obtained by rotating the Top Left Corner Templates clockwise by 90 degrees. As shown in Figure 6.7, many ‘false corners’ were detected in the video image, these points often correspond to objects surrounding the building and are not present in the 3D model, such as trees and signs. To narrow the search space for correspondence, detected corners are rejected if they do not meet a condition that forms a Vertical Line Feature. That is, all of the following conditions should be met:

1) A Left/Right Vertical Line Feature is a continuous line of pixels that connect a Top Left/Right Corner to a Bottom Left/Right Corner.

2) Given the camera is kept level by the gimbal system, the slope of the line should be within a tolerable range of the vertical defined by the image's vertical axes.
3) The Top Left/Right Corner should be higher in the image than the Bottom Left/Right Corner.

All the combinations of Top Left/Right Corners and Bottom Left/Right Corners are sequentially tested against the various conditions. If a Top Left/Right Corner and Bottom Left/Right Corner do not meet a condition, the pair of points is discarded and the next combination of points is tested. Tests are performed in increasing order according to computational complexity to increase efficiency. Figure 6.7 shows Left (red) and Right (green) Vertical Line Features extracted from the sample UAV image. Using the same method, Vertical Line Features were extracted from the synthetic image. The 3D mode's vertices were classified according to corner type, and vertical edges were classified as either Left or Right Vertical Lines. Figure 6.8 shows the Left Vertical Line Features (red) and Right Vertical Line Features (green) that are visible in the synthetic image. Top and Bottom Horizontal Lines Features are extracted in a similar fashion. A Top Horizontal Line is composed of a Top Left Corner and Top Right Corner, connected by a horizontal image line. Likewise, a Bottom Horizontal Line is composed of a Bottom Left Corner and Bottom Right Corner, connected by a horizontal image line.

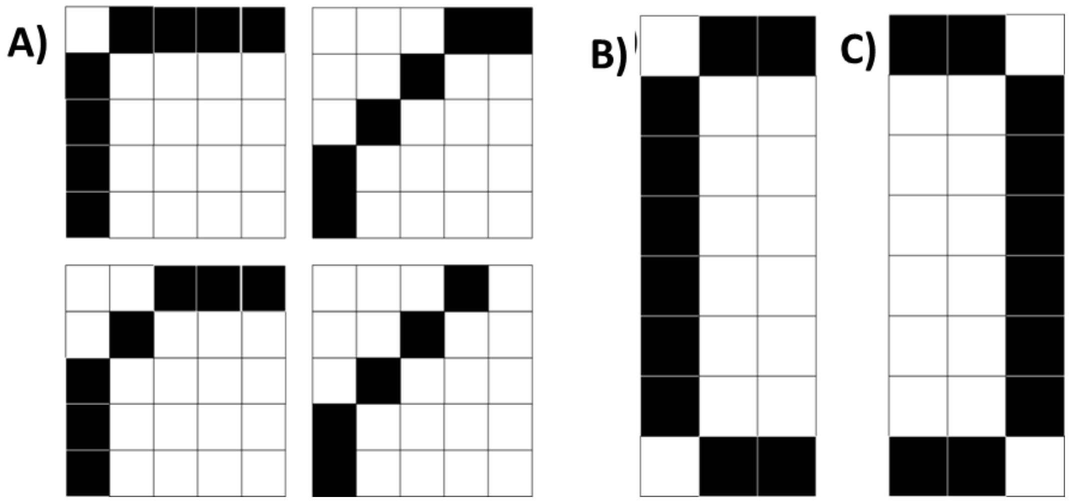


Figure 6.6. (A) Top Left Corner templates, (B) Left Vertical Line Feature, (C) Right Vertical Line Feature.

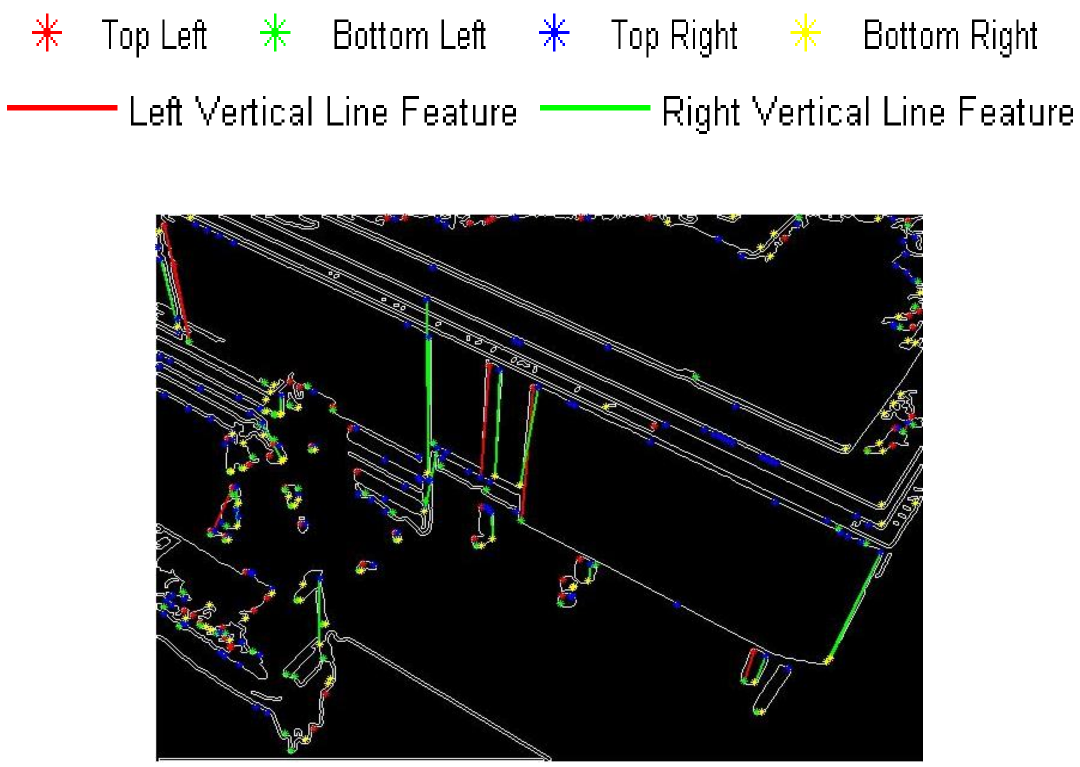


Figure 6.7. The extracted corners and Vertical Line Features from the edge image of the video frame.

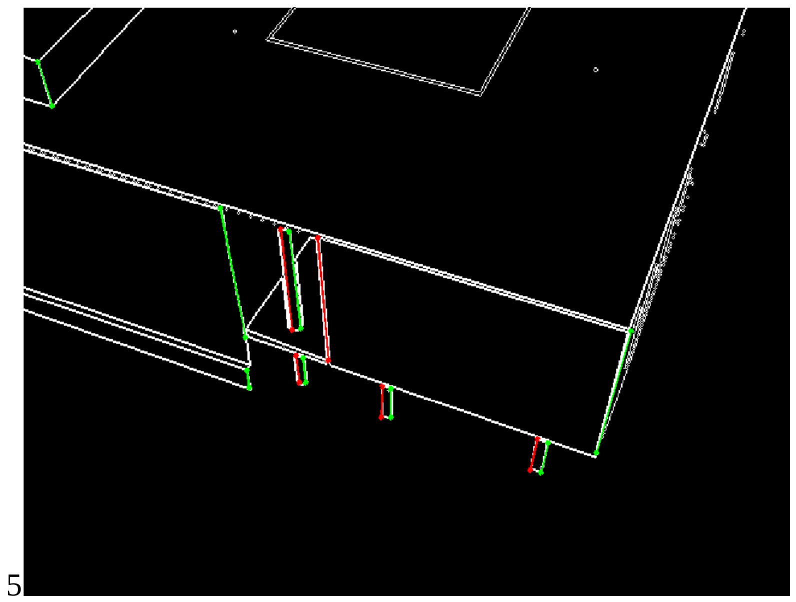


Figure 6.8. Left (red) and Right (green) Vertical Line Features extracted from the synthetic image.

6.4 Narrowing the search space in feature matching

Given the scales of the synthetic image and the video frame are similar, the horizontal spacing between the vertical lines should be similar as well. Thus, a horizontal shift should align both sets of vertical lines. This shift is evident in Figure 6.9, where for each vertical line, the column value of the line's centroid is plotted against the length of the same line. The red series signifies the lines extracted from the video frame, and the green series corresponds to the lines from the synthetic image. The triangles symbolize Left Vertical Line Features, and squares symbolize Right Vertical Line Features.

The maximum value of the cross correlation of the red (video) series and green (model) series provides the numerical value of the horizontal shift. The vertical lines from the synthetic image are shifted and matched with the vertical lines from the video frame based on proximity and line length. Figure 6.9 shows the lines from the video images with the shifted lines from the synthetic image. Potential Vertical Line Feature matches are circled in Figure 6.10. Matching Vertical Line Features via correlation between the video image and synthetic image has been successful with the orientation angles differing by as much as 11 degrees.

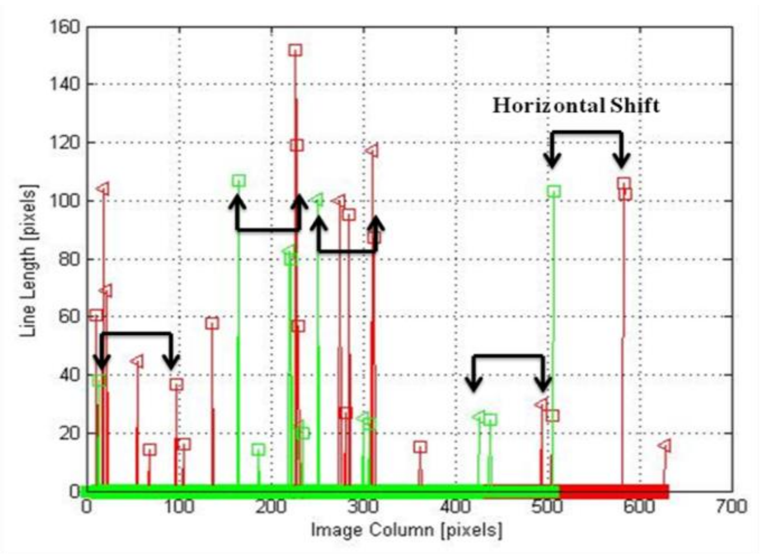


Figure 6.9. Horizontal shift between the Vertical Line Features of the video frame (red) and synthetic image (green).

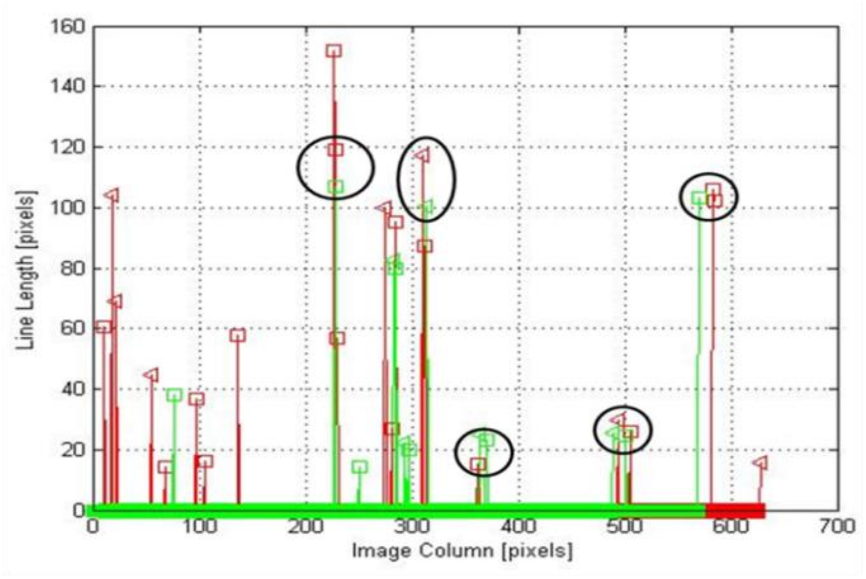


Figure 6.10. Corresponding Vertical Line Features between the video frame (red) and synthetic image (green) are circled.

6.5 Pose estimation via space resection

Once correspondence between the sensor data and the map is established, through either absolute or incremental techniques, different forms of space resection may be used to estimate the camera pose from the corresponding features. For example, VVS uses a point-based space resection. By using a line-based space resection, it is not necessary to identify congruent points in the image and the 3D model. Of the major works covering the concept of implementing straight lines as an alternative or a complementary to point features, one could refer to the orientation of images on the basis of indirect lines (Mulawa and Mikhail, 1988), space resection, and image orientation based on coplanarity equations (Habib et al. (2003), van den Heuvel (1999)), image orientation using collinearity equations (Tommaselli and Lugnani (1998), Schenk (2004), Habib and Kelley (2001)), aerial triangulation (Habib et al. (2000), Habib et al. (2002)), and determination of the calibration parameters of aerial cameras (Brown (1971), (Habib et al. (2002)), as well as coreferencing (Shaker (2004), Habib et al. (2004), Habib and Alruzouq (2004)) and the geometrical correction of satellite images (Long et al., 2011). Nonetheless, no research has been accomplished, to the author's knowledge, about the utilization of linear features in real-time

UAV-based photogrammetry. The following sections describe the pose estimation strategies used in this dissertation.

6.5.1 Point-based image orientation

A single photo resection is used to determine the interior and exterior orientation parameters of an image. The exterior orientation parameters (EOPs) includes the orientation angles (ω, ϕ, *κ*) and the position (*X₀*, *Y₀*, *Z₀*) of the perspective centre in relation to the object reference system. The EOPs are also referred to in literature as camera pose. The interior orientation parameters (IOPs) includes the camera's focal length (*f*), principal point coordinates (x_0, y_0), and radial lens distortion (*dx*, *dy*). The resection is accomplished by measuring the image coordinates of object features. When these object features have known 3D coordinates in the object reference system, they are known as control points. The applied collinearity model (Equation (6.1)) describes the projection of an object point onto the image plane of a camera:

(6.1)

$$x = x_0 - f * \frac{r_{11} * (X - X_0) + r_{21} * (Y - Y_0) + r_{31} * (Z - Z_0)}{r_{13} * (X - X_0) + r_{12} * (Y - Y_0) + r_{13} * (Z - Z_0)} + dx$$

$$y = y_0 - f * \frac{r_{12} * (X - X_0) + r_{22} * (Y - Y_0) + r_{32} * (Z - Z_0)}{r_{13} * (X - X_0) + r_{12} * (Y - Y_0) + r_{13} * (Z - Z_0)} + dy$$

where r_{ij} = elements of rotation matrix

x, y = image coordinates

X, Y, Z = object coordinates

In this work, the lens distortion is limited to the radial distortion, Brown (1971) provides the radial lens distortion correction as:

(6.2)

$$dx = x' * k_1 * \left(r'^2 - r_0^2\right) + k_2 * \left(r'^4 - r_0^4\right) + k_3 * (r'^6 - r_0^6)$$

$$dy = y' * k_1 * \left(r'^2 - r_0^2\right) + k_2 * \left(r'^4 - r_0^4\right) + k_3 * (r'^6 - r_0^6)$$

$$r' = \sqrt{x'^2 + y'^2}$$

where r_0 = sensor specific parameters (zero-distortion radius)

x', y' = image coordinates related to the principal point

At least three non-collinear control points are required to calculate the exterior orientation parameters. A non-linear least squares adjustment is possible with the use of more points. The over-determined system of equations is solved by iteratively minimizing the sum of the squared deviations between the projected object features and their corresponding image measurements. If a sufficient number of suitably distributed control points are available, the parameters of the interior orientation and of the lens distortion can be estimated simultaneously. Estimating the interior orientation is mainly useful for photos taken with a non-metric camera because the focal length and principal point may vary throughout the images.

6.5.2 Line-based image orientation

Kubik (1991) presented an approach to perform a single photo resection using straight 3D lines in the object space and the measurement of points on the corresponding lines in the image, in other words point-to-line correspondences. Schwermann (1995) presented an approach to perform a single photo resection using straight 3D lines in the object space and the measurement of corresponding lines in the image, in other words line-to-line correspondences. The following sections describe both the point-to-line resection and the line-to-line, respectively.

6.5.2.1 3D line representation

There are different methods for describing straight lines in Euclidean 3D space. Here, a description with four parameters is chosen, which was published by Roberts (1988) and resumed and varied by Schenk (2004). This line representation was chosen because it is simple and free of singularities, so no special cases have to be considered. Further, the number of parameters used to represent the line is equal to the degree of freedom of a 3D line (i.e., 4). The two positional parameters (X_s, Y_s) and the orientation parameters (α, θ) are visualized in Figure 6.11.

The azimuth α and the zenith angle θ can be calculated from the spherical coordinates of the direction vector **d** of the line (Figure 6.11 (left)). A unique representation of the line results when the angles are limited to $0 < \alpha < 360°$ and $0 < \theta < 90$, i.e., the line is restricted to the hemisphere of the positive Z-axis. The following matrix describes the rotation about the Z-axis by angle α and the subsequent rotation about the new Y-axis (i.e., Y') by angle θ.

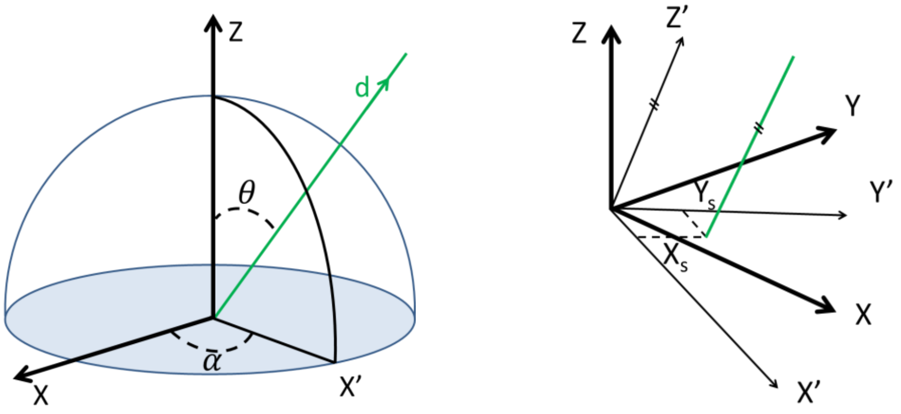


Figure 6.11. The four-parameter representation of a line in object space: Two positional parameters define the line's intersection with a plane perpendicular to it (X_s, Y_s) and two orientation parameters define the direction of line (α, θ) (Meierhold et al., 2008).

(6.3)

$$R_{\alpha\theta} = \begin{bmatrix} \cos\alpha * \cos\theta & \sin\alpha * \cos\theta & -\sin\theta \\ -\sin\alpha & \cos\alpha & 0 \\ \cos\alpha * \sin\theta & \sin\alpha * \sin\theta & \cos\theta \end{bmatrix}$$

The resulting X' and Y' axis reside on a plane which passes through the origin and is perpendicular to the line. The two positional parameters result from the intersection of the line with this plane (Figure 6.11 (right)). The intersection point corresponds to the closest point on the line to the origin.

(6.4)

$$\boldsymbol{p}' = \begin{pmatrix} X_s \\ Y_s \\ t \end{pmatrix} = \boldsymbol{R}_{\alpha\theta} * \boldsymbol{p}$$

The rotation of any point **p**=(X, Y, Z)^T on the line from the object coordinate system to the new coordinate system (Equation (6.4)) leads to the same planimetric coordinates (X_s, Y_s), but to different coordinates along the Z' axis expressed by the parameter t. The equation of the line in the object space can be recovered by inversion of Equation (6.4):

(6.5)

$$\boldsymbol{p} = \boldsymbol{R}_{\boldsymbol{\alpha\theta}}^{\boldsymbol{T}} * \boldsymbol{p'} = \begin{pmatrix} X_s * \cos\alpha * \cos\theta - Y_s * \sin\alpha + \mathrm{t} * \cos\alpha * \sin\theta \\ X_s * \sin\alpha * \cos\theta + Y_s * \cos\alpha + \mathrm{t} * \sin\alpha * \sin\theta \\ -X_s * \sin\theta + \mathrm{t} * \cos\theta \end{pmatrix}$$

6.5.2.2 The point-to-line space resection

The 3D lines are described by 4 parameters (X_s, Y_s, α, θ) as presented in Section 6.5.2.1. In the physical interpretation, rays projecting from the camera perspective centre through the image points should intersect the corresponding object lines (Figure 6.12). The point-to-line correspondence can be described mathematically by modifying the collinearity equations. Where the object coordinates (X, Y, Z) are replaced with the respective row of the line equation in Equation (6.5):

(6.6)

$$\text{with } \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{pmatrix} X_s * \cos\alpha * \cos\theta - Y_s * \sin\alpha + \mathrm{t} * \cos\alpha * \sin\theta \\ X_s * \sin\alpha * \cos\theta + Y_s * \cos\alpha + \mathrm{t} * \sin\alpha * \sin\theta \\ -X_s * \sin\theta + \mathrm{t} * \cos\theta \end{pmatrix}$$

As in the pure point-based method, the parameters of the exterior orientation (X_0, Y_0, Z_0, ω, ϕ, κ) are estimated in an iterative process, where the residuals of the observed image coordinates are minimized. Additionally, one degree of freedom is added for every point measurement as the line parameter t has to be estimated for every image point. When measuring two image points per object line, at least three known non-parallel object lines are required for determination of the exterior orientation: 6 unknown EOPs + 6 unknowns line parameters (t) = 12 unknowns; 6 point measurements x 2 image coordinate measurements per point = 12 measurement equations. Notably, the EOPs estimated by the point-to-line resection are identical to the EOPs estimated by the point-to-point resection. The increase in measurement redundancy from the point-to-line to the point-to-point resection yields a respective decrease in estimated EOP standard deviation.

The parameters of the interior orientation (x_0, y_0, f) as well as the parameters of the radial distortion (k_1, k_2, k_3) can be calculated from at least six known spatially distributed object lines (e.g., in planes with different levels of depth) with two corresponding image points. For increasing the redundancy and consequently improving the accuracy of the adjustment, the number of image points per object line can be increased.

Notably, the point-to-line resection yields 3D object coordinates of image measurements that are not necessarily present in the 3D model. As a result, this technique could be considered as a map-building algorithm.

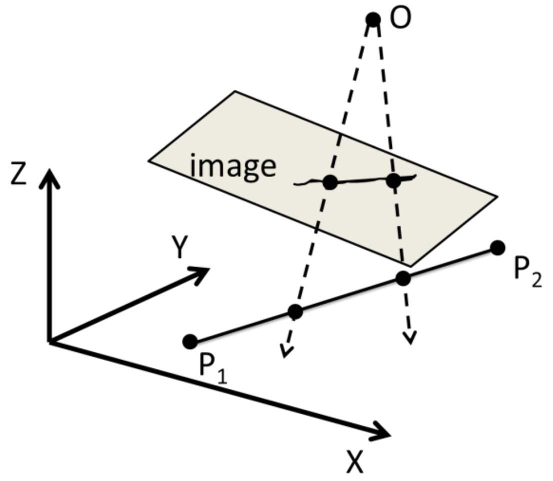


Figure 6.12. Point-to-line collinearity approach to establish relationships between object space lines and their partial projections in images (Meierhold et al., 2008).

6.5.2.3 The line-to-line space resection

The equations of the image lines are defined as:

$$y = m * x + n \tag{6.7}$$
$$x = m' * y + n'$$

where m, m' = slope

n, n' = intersection with appropriate axis

Schwermann (1995) used this type of line equations for the description of the object lines as well. However, this representation is only applicable for straight lines with small slopes. To avoid special cases with respect to appropriate axes, Meierhold et al. (2008) modified the method by applying the 4-parameter line equations presented in Section 6.5.2.1.

This line-to-line correspondence can be described mathematically by modification of the collinearity equations in such a way that the image line parameters (m, n) are dependent on the parameters of the object line (X*s, Ys, α, θ*) and the parameters of the exterior and interior orientation. The unknown line parameter *t* was eliminated because points are no longer considered. The resulting equations are shown below:

(6.8)

$$m = \frac{V_2 * W_1 - V_1 * W_2}{U_2 * W_1 - U_1 * W_2}$$

$$n = y_0 - x_0 * \frac{V_2 * W_1 - V_1 * W_2}{U_2 * W_1 - U_1 * W_2} + c * \frac{U_1 * V_2 - U_2 * V_1}{U_2 * W_1 - U_1 * W_2}$$

where

(6.9)

$$U_1 = r_{11} * \cos\alpha * \sin\theta + r_{21} * \sin\alpha * \sin\theta + r_{31} * \cos\theta$$

$$V_1 = r_{12} * \cos\alpha * \sin\theta + r_{22} * \sin\alpha * \sin\theta + r_{32} * \cos\theta$$

$$W_1 = r_{13} * \cos\alpha * \sin\theta + r_{23} * \sin\alpha * \sin\theta + r_{33} * \cos\theta$$

$$U_2 = r_{11} * (A - X_0) + r_{21} * (B - Y_0) + r_{31} * (C - Z_0)$$

$$V_2 = r_{12} * (A - X_0) + r_{22} * (B - Y_0) + r_{32} * (C - Z_0)$$

$$W_2 = r_{13} * (A - X_0) + r_{23} * (B - Y_0) + r_{33} * (C - Z_0)$$

with

(6.10)

$$A = X_s * \cos\alpha * \cos\theta - Y_s * \sin\alpha$$

$$B = X_s * \sin\alpha * \cos\theta - Y_s * \cos\alpha$$

$$C = -X_s * \sin\theta$$

In the special case of a vertical image line, the inverse relations of Equation (6.8) have to be considered as observations in the non-linear adjustment process.

$$m' = \frac{1}{m}$$
$$n' = -\frac{n}{m} \tag{6.11}$$

The exterior orientation parameters of an image can be estimated from at least three straight lines defined in the object space. The observation model does not include lens distortion because the correction model in Equation (6.1) is not applicable. Thus, a distortion model that is dependent on the image line (m, n) has to be developed. Alternatively, the lens distortion can be calibrated and removed from the image before the image orientation process. The measurement of the image lines would then be performed in the resampled image. Meierhold et al. (2008) suggested an integrated orientation procedure using elements from plumb-line calibration (Lerma and Cabrelles, 2007). Brown (1971) first developed the plumb line method of self-calibration in close-range photogrammetry. Using images containing straight lines, Brown evaluated the extent of the deviation of these lines from a straight direction to estimate the radial and decentring distortion of the lens, as well as the coordinates of the principal point. Notably, it has been demonstrated that it is possible to use straight lines for camera calibration (i.e., estimation of image interior orientation parameters) for both frame-type (van den Heuvel (1999), Wang and Tsai (1990)) and push-broom (Habib, et al., 2003) images. Most recently, Babapour et al. (2017) presented a Jacobi–Fourier (JF) combined orthogonal model that used for self-calibration of digital cameras using a standalone post-mission process. They showed an improvement in correcting the complex camera distortions compared with the traditional self-calibration methods (Babapour et al., 2016). This is particularly important in UAV-based photogrammetry, where images derived from non-metric cameras require more advanced manipulations.

6.6 Results: A novel map-based incremental localization algorithm

The following experiments demonstrate two novel applications of the proposed model-based tracker:

1) A UAV localization solution for bridging GNSS gaps and aiding other navigation sensors (GNSS/INS) using only one RGB camera.
2) An indoor localization solution for UAVs using only one RGB-D camera.

The results show that in terms of accuracy and tracking maintenance, the proposed model-based tracker demonstrated a significant improvement compared to the state-of-the-art. In both the outdoor and indoor experiments, a value of one quarter (¼) of the image width (120 pixels given a 640 x 480 pixel (480p) image resolution) was used for all three thresholds (Table 6.1). This threshold was chosen because it reduced the number of candidate lines to an adequate amount for real-time matching onboard the UAV, while yielding lines with adequate length to estimate the camera pose with a precision less than 3 times the geometric accuracy of the 3D model. The pose estimated from the point-to-point resection was accepted if the estimated 3D positional standard deviation was less than 50 cm in the outdoor environment and 10 cm in the indoor environment. These threshold values were chosen based on the geometric accuracy of the 3D model used in the experiment.

Table 6.1. Threshold values used in the outdoor and indoor experiments.

Threshold parameter	**Threshold value**
Line length	¼ image width
Vertical line separation	¼ image width
Horizontal line separation	¼ image width
3D standard deviation (outdoor)	50 cm
3D standard deviation (indoor)	10 cm

6.6.1 Experiment 1: Map-based incremental localization in outdoor environments

This experiment assessed the proposed model-based tracker in tracking a georeferenced 3D building model using the RGB camera of a UAV for the purpose of UAV navigation in GPS-denied outdoor environments. Previous model-based trackers have been demonstrated to perform well in tracking small, simple objects in small and cluttered indoor environments. This section demonstrates a novel model-based tracker capable of tracking larger, complex objects (i.e., building models) in large outdoor environments by UAVs. This experiment specifically aimed to show improvement, in terms of accuracy and tracking maintenance, from the results obtained in Section 5.5, where improper model projection and maintained misalignment between the image and the model caused larger than expected average 3D positional standard deviations (11.07 m ± 15.51 m) and lost tracking.

The 3D virtual building model of York University's Keele Campus (Armenakis and Sohn, 2009) was used as a known environment, this is the same model used in the experiment presented in Section 5.5. The 3D CAD model served two purposes in the proposed approach. Firstly, it provided the necessary level of detail such that individual buildings could be uniquely identified by the model-based tracker. Secondly, it provided ground control points to photogrammetrically estimate the camera pose.

The same 1137 frames that were processed in the experiment presented in Section 5.5 were processed in this experiment. The camera's intrinsic parameters were calibrated beforehand and held fixed in the pose estimation process. The image frames were tested at 480p, 720p, and 1080p resolutions. Trials revealed that, at 40 metres above ground level, the 480p video performed the best in terms of processing speed and tracking performance. The tracking performance improved because the noisy edges detected at higher resolutions were removed at lower resolutions, leaving only the stronger edge responses. The following sections provide a quantitative analysis of the obtainable precision and accuracy of the localization algorithm in outdoor environments.

6.6.1.1 Precision assessment

Figure 6.13 shows the five frames from the outdoor dataset where the drift correction occurred. The top row shows the real image captured by the UAV's camera, below each image is its corresponding synthetic image. The red lines show the two horizontal lines (labelled 1 and 2) and two vertical lines (labelled 3 and 4) that were matched using line-to-line space resection. Table 6.2 shows the average and standard deviation of the pose standard deviations from the least squares adjustment for the 5 sample frames in Figure 6.13.

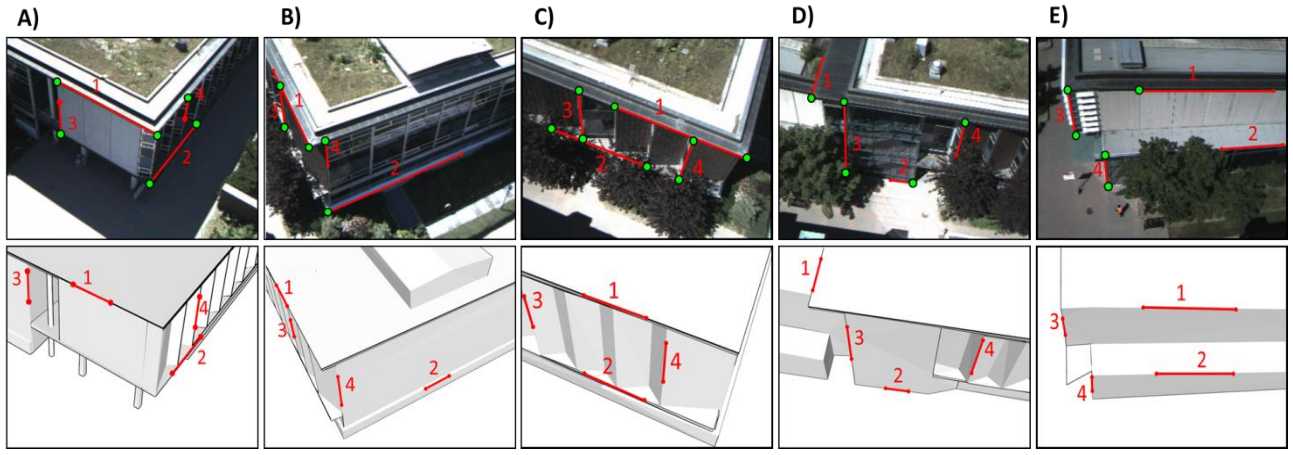


Figure 6.13. Outdoor experiment: Tracking a 3D building model using the camera of a UAV.

Table 6.2. Outdoor experiment: Average ($\bar{\sigma}$) and standard deviation (σ_σ) of the pose standard deviations from the least squares adjustment for the 5 sample frames in Figure 6.13.

	$\overline{\sigma_X} \pm \sigma_{\sigma_X}$ [m]	$\overline{\sigma_Y} \pm \sigma_{\sigma_Y}$ [m]	$\overline{\sigma_Z} \pm \sigma_{\sigma_Z}$ [m]	$\overline{\sigma_{3D}} \pm \sigma_{\sigma_{3D}}$ [m]	$\overline{\sigma_\omega} \pm \sigma_{\sigma_\omega}$ [°]	$\overline{\sigma_\varphi} \pm \sigma_{\sigma_\varphi}$ [°]	$\overline{\sigma_k} \pm \sigma_{\sigma_k}$ [°]
L2L	1.980 ± 1.755	1.0257 ± 0.493	1.155 ± 0.698	2.511 ± 1.951	1.765 ± 0.762	2.517 ± 2.394	1.036 ± 0.440
P2L	1.264 ± 1.319	1.077 ± 0.600	0.885 ± 0.375	1.881 ± 1.496	1.706 ± 0.650	1.661 ± 1.774	0.974 ± 0.570
P2P	0.219 ± 0.049	0.3322 ± 0.067	0.252 ± 0.020	0.471 ±0.085	0.476 ± 0.110	0.321 ± 0.089	0.267 ± 0.100

Table 6.3. Outdoor experiment's check point statistics: The averages and standard deviations of the differences between ground truth model coordinates and object coordinate calculated by the point-to-line resection for horizontal lines and vertical lines.

	Number of lines	$\overline{\Delta X} \pm \sigma_{\Delta X}$ [m]	$\overline{\Delta Y} \pm \sigma_{\Delta Y}$ [m]	$\overline{\Delta Z} \pm \sigma_{\Delta Z}$ [m]	$\overline{\Delta 3D} \pm \sigma_{\Delta 3D}$ [m]
Horizontal lines	19	0.01 ± 0.24	-0.221 ± 0.44	0.026 ± 0.221	0.222 ± 0.547
Vertical lines	21	0.001 ± 0.03	-0.006 ± 0.05	-0.117 ± 1.065	0.117 ± 1.067

As expected, the line-to-line (L2L) space resection yielded the least accurate pose estimates, followed by the point-to-line (P2L) space resection. The point-to-point (P2P) space resection provided the most accurate estimates. 46 cm 3D positioning accuracies were achieved from the P2P space resection. This is acceptable considering the geometric accuracy of the 3D model is 10 to 40 cm.

The green circles in Figure 6.13 indicate check points. The 3D object coordinates of these points were estimated by the point-to-line resection and compared with the ground truth coordinates (manually extracted from the model's vertices). There were 19 check points from horizontal lines and 21 check points for vertical lines for outdoor scenario. Table 6.3 provides the check point statistics. The Z component of the vertical lines was the noisiest (± 1.065 m). This could be due to the distribution of vertical lines across the images, outliers, or inaccuracies of the 3D model. The average 3D difference from the point cloud to the 3D (0.222 ± 0.547 m and 0.117 ± 1.067m for horizontal and vertical lines, respectively) are within the accuracies of the 3D model (10 to 40 cm), suggesting that the proposed tracker effectively identified corresponding features and rejected outlier matches.

6.6.1.2 Accuracy assessment

The 1137 video images captured in the outdoor dataset presented in Section 5.5.3 were processed with VisualSFM and SURE (Figure 6.14). The reconstruction system integrated the map-building incremental localization algorithms described in Section 5.2. Explicitly, a GPU-based SIFT module parallelized matching, a multicore and GPU-based SfM module estimated the camera parameters and generated a sparse point cloud, and the PMVS/CMVS/SURE tool chain efficiently densified the sparse point cloud. The reconstruction system resulted in one point cloud, implying that incremental tracking was never lost throughout the image sequence.

The camera poses estimated from the proposed model-based tracker were used to transform the photogrammetrically derived 3D point cloud to the 3D model's coordinate system (Figure 6.15). Check points were used to assess the accuracy of the georeferenced 3D point cloud generated from the UAV's sensor data (Figure 6.16). The ground truth coordinates were extracted from the

3D building model. The results from manually comparing corresponding points between the UAV’s point cloud and building model are in Table 6.4.

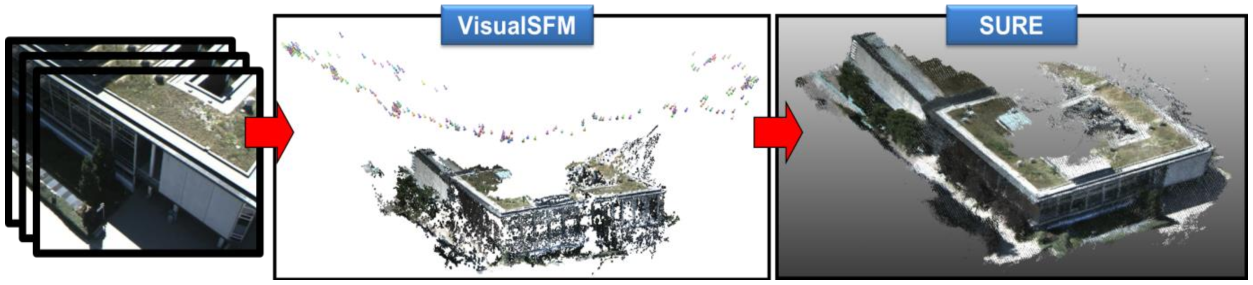


Figure 6.14. VisualSFM and SURE used to generate a dense point cloud of the Lassonde Building from the UAV's video imagery.

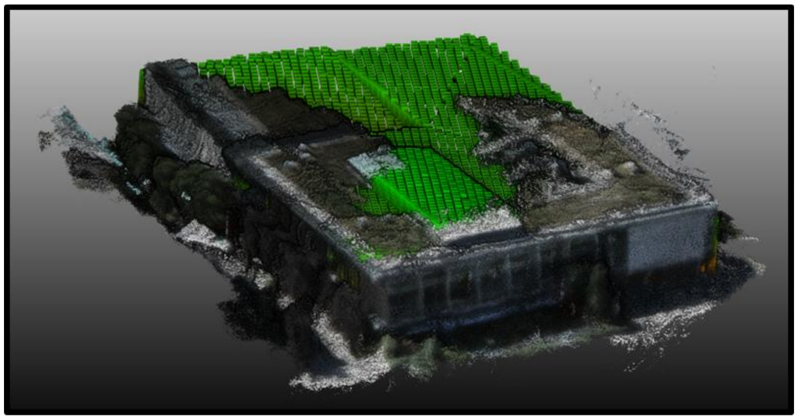


Figure 6.15. Point cloud registered to the 3D using the pose estimated by the proposed MBT.

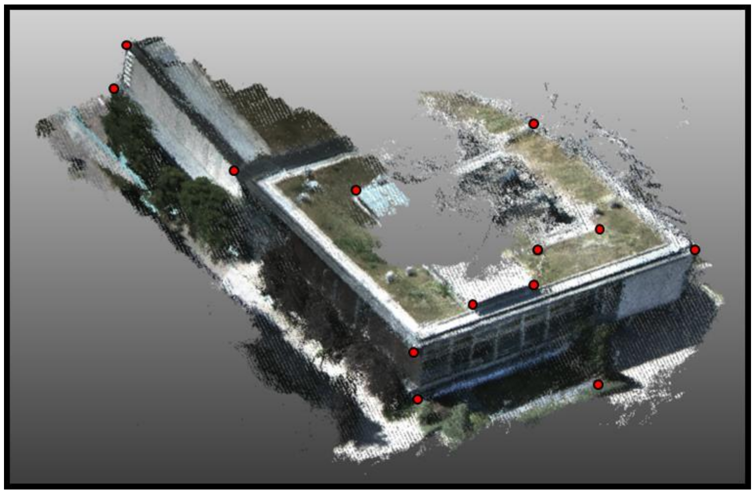


Figure 6.16. Check points (red circles) for accuracy assessment of the point cloud.

Registering the UAV's point cloud with the 3D building model resulted in 56 cm mapping accuracies (root mean square error (RMSE)). The registration accuracy was also evaluated using the CloudCompare (www.danielgm.net/cc/) software. The Multiscale Model to Model Cloud Comparison (M3C2) algorithm (Lague et al., 2013) was used to statistically detect significant changes between the UAV's point cloud and a point cloud sampled from the model. M3C2 computes the local distance between two point clouds along the normal surface direction. Distances larger than a user defined threshold were considered outliers. The estimated points not belonging to the model were segmented based on this threshold. They mainly consisted of vegetation surrounding the building (white points in Figure 6.17). A threshold value of 0.56 m was used because this was the estimated noise ($RMSE_{3D}$) calculated from the check points (Table 6.4).

Table 6.4. Accuracy assessment of the generated point clouds

Building	**# Check points**	**$RMSE_{3D}$ [m]**	**Average 3D error [m]**
Lassonde	13	0.56	0.61

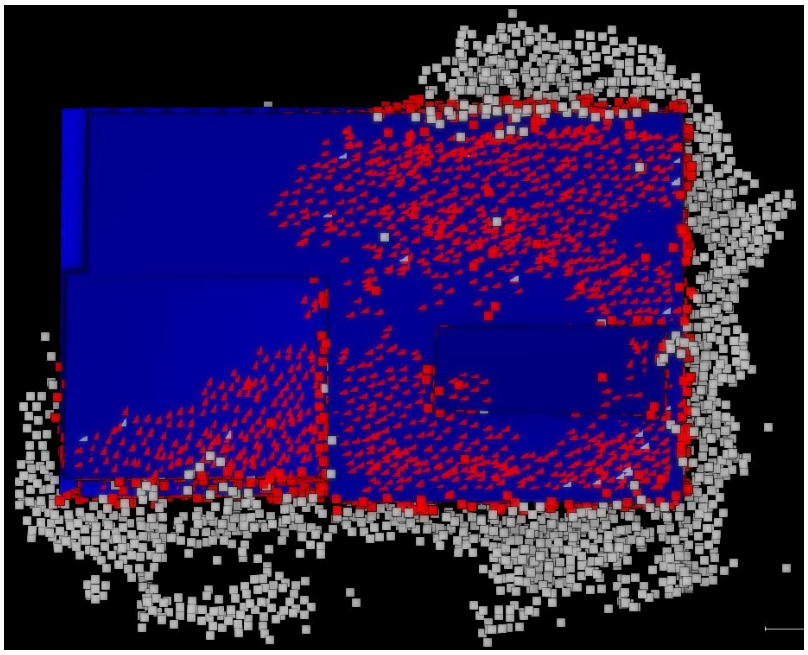


Figure 6.17. Detecting the estimated points belonging (red) and not belonging (white) to the model.

The remaining points (red points in Figure 6.17) were used to evaluate the registration quality between the UAV's point cloud and the building model (Figure 6.18). The average point to point difference between UAV's point cloud and the model for the Lassonde Building was -23.1cm ±43.2cm. This was acceptable given the geometric accuracy of the 3D model is 10 to 40 cm. It is evident from Figure 6.18 that the larger differences correspond to vegetation points that still remain after the filtering process.

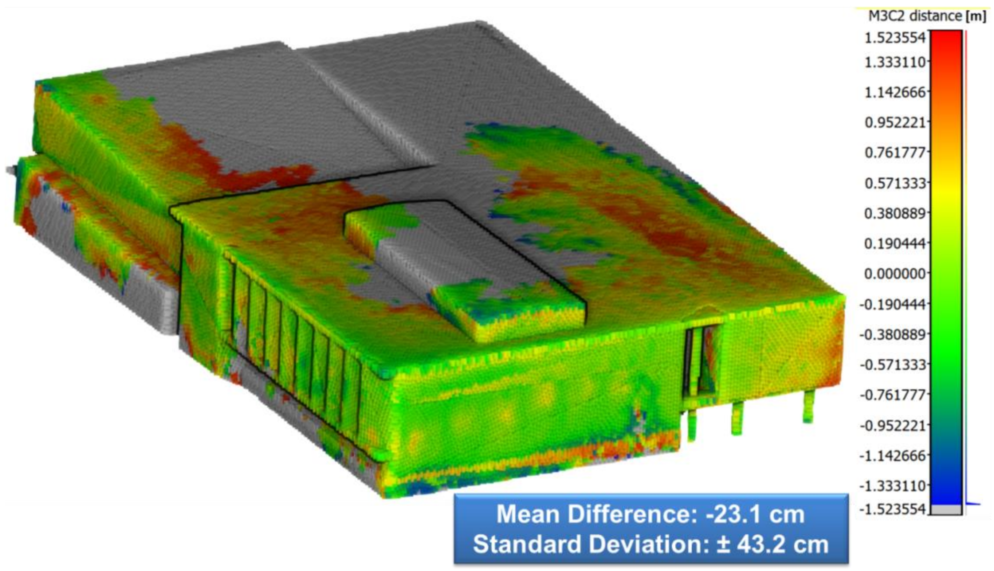


Figure 6.18. Registration quality between the UAV's point cloud and the model.

6.6.2 Experiment 2: Map-based incremental localization in indoor environments

The second experiment assessed the proposed model-based tracker in tracking a 3D indoor model using the RGB-D camera of a UAV for the purpose of UAV navigation in GNSS-denied indoor environments. The experiment was performed on the second floor of York University's Bergeron Centre for Engineering Excellence, the same model that was used in the experiments presented in Section 5.5. The experiment began with the user providing an initial approximation of the camera pose. Corresponding straight lines between the image sequence and the 3D indoor model were then automatically identified. At each frame, the corresponding lines were used in a space resection to estimate the camera pose. Figure 6.19 shows the trajectory of the UAV within the 3D CAD model of the building with stand-alone RGB-D SLAM (left), and with the proposed model-based tracker (MBT) integrated with RGB-D SLAM (right). Notably, RGB-D SLAM was functioning as a visual odometer, that is, loop closure detection was disabled. It is evident that the drift present in the stand-alone RGB-D SLAM solution is mitigated by the proposed model-based tracker. The following sections provide a quantitative analysis of the obtainable precision and accuracy of the localization algorithm in indoor environments.

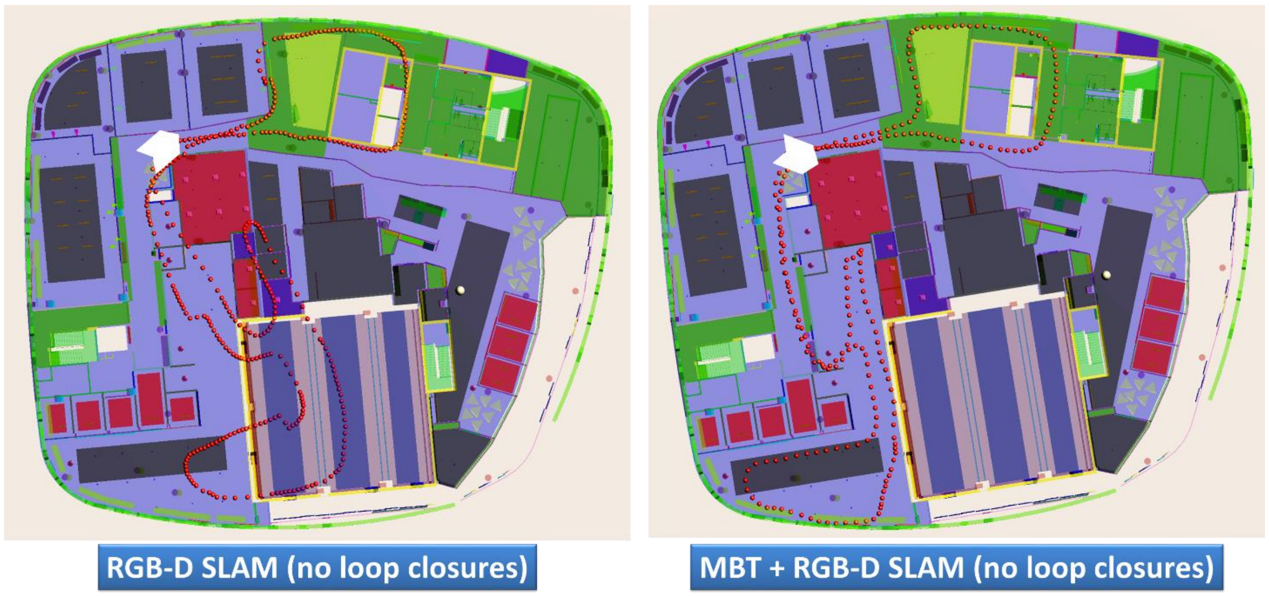

RGB-D SLAM (no loop closures)
MBT + RGB-D SLAM (no loop closures)

Figure 6.19. The trajectory of the UAV (red circles) in the indoor environment without model-based tracking (left) and with model-based tracking (right).

6.6.2.1 Precision assessment

Figure 6.20 shows five frames from the indoor dataset where the drift correction occurred. The top row shows the real image captured by the UAV's camera, below each image is its corresponding synthetic image. The red lines show the two horizontal lines (labelled 1 and 2) and two vertical lines (labelled 3 and 4) that were matched using line-to-line space resection.

Table 6.5 show the average and standard deviation of the pose standard deviations from the least squares adjustment for the 5 sample frames in Figure 6.20. As expected, the line-to-line (L2L) space resection yielded the least accurate pose estimates, followed by the point-to-line (P2L) space resection. The point-to-point (P2P) space resection provided the most accurate estimates.

1.4 cm 3D positioning accuracies were achieved from the P2P space resection. This was acceptable considering the geometric accuracy of the 3D model is 2 to 5 cm.

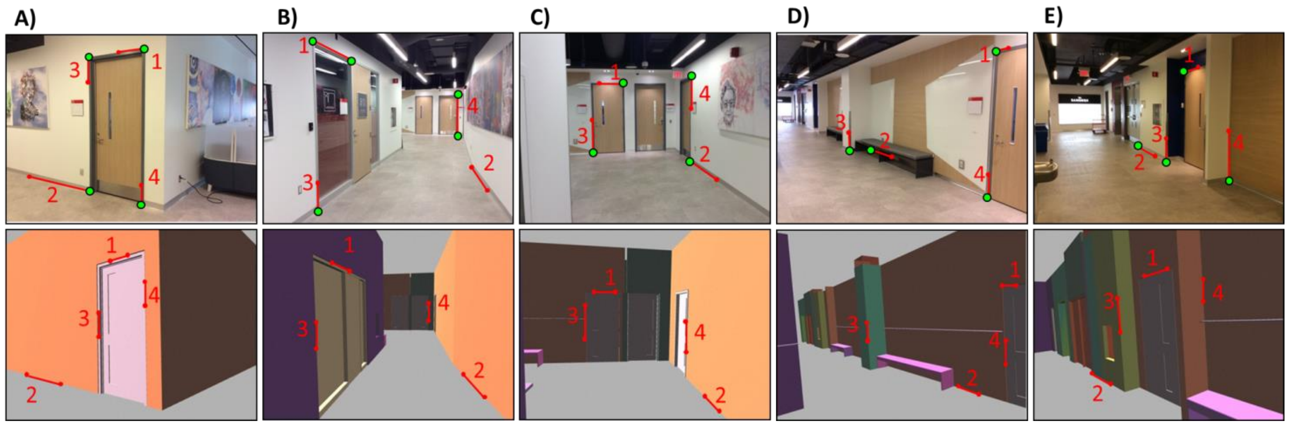


Figure 6.20. Indoor experiment: Tracking a 3D indoor model using the camera of a UAV.

Table 6.5. Indoor experiment: Average ($\bar{\sigma}$) and standard deviation (σ_σ) of the pose standard deviations from the least squares adjustment for the 5 sample frames in Figure 6.20.

	$\overline{\sigma_X} \pm \sigma_{\sigma_X}$ [m]	$\overline{\sigma_Y} \pm \sigma_{\sigma_Y}$ [m]	$\overline{\sigma_Z} \pm \sigma_{\sigma_Z}$ [m]	$\overline{\sigma_{3D}} \pm \sigma_{\sigma_{3D}}$ [m]	$\overline{\sigma_\omega} \pm \sigma_{\sigma_\omega}$ [◦]	$\overline{\sigma_\varphi} \pm \sigma_{\sigma_\varphi}$ [◦]	$\overline{\sigma_k} \pm \sigma_{\sigma_k}$ [◦]
L2L	0.127 ± 0.123	0.127 ± 0.127	0.149 ± 0.120	0.233 ± 0.213	2.416 ± 2.576	1.435 ± 1.205	1.671 ± 1.905
P2L	0.085 ± 0.042	0.102 ± 0.069	0.093 ± 0.027	0.162 ± 0.085	1.831 ± 1.298	1.048 ± 0.601	1.540 ± 1.342
P2P	0.007 ± 0.005	0.006 ± 0.003	0.011 ± 0.007	0.014 ± 0.009	0.240 ± 0.182	0.100 ± 0.081	0.181 ± 0.143

The green circles in Figure 6.20 indicate check points. The 3D object coordinates of these points were estimated by the point-to-line resection and compared with the ground truth coordinates (manually extracted from the model's vertices). There were 9 check points from horizontal lines and 10 check points for vertical lines for outdoor scenario. Table 6.6 provides the check point statistics. The Z component of the vertical lines was the noisiest. This could be due to the distribution of vertical lines across the images, outliers, or inaccuracies of the 3D model. The standard deviations are within the accuracies of their respective 3D models, suggesting that the proposed tracker effectively identified corresponding features and rejected outlier matches.

The Z component of the vertical lines was the noisiest (± 0.090 m). This could be due to the distribution of vertical lines across the images, outliers, or inaccuracies of the 3D model. The average 3D difference from the point cloud to the 3D (0.007 ± 0.043 m and 0.011 ± 0.093m for horizontal and vertical lines, respectively) are within the accuracies of the 3D models (0.02 to 0.05 m), suggesting that the proposed tracker effectively identified corresponding features and rejected outlier matches.

Table 6.6. Indoor experiment's check point statistics: The averages and standard deviations of the differences between ground truth model coordinates and object coordinate calculated by the point-to-line resection for horizontal lines and vertical lines.

	Number of lines	$\overline{\Delta X} \pm \sigma_{\Delta X}$ **[m]**	$\overline{\Delta Y} \pm \sigma_{\Delta Y}$ **[m]**	$\overline{\Delta Z} \pm \sigma_{\Delta Z}$ **[m]**	$\overline{\Delta 3D} \pm \sigma_{\Delta 3D}$ **[m]**
Horizontal lines	9	-0.007 ± 0.04	0.001 ± 0.017	-0.001 ± 0.005	0.007 ± 0.043
Vertical lines	10	-0.001 ± 0.01	-0.009 ± 0.022	0.007 ± 0.090	0.011 ± 0.093

6.6.2.2 Accuracy assessment

RGB-D SLAM was initialized in the 3D model's coordinate system, as opposed to a local (odometry) coordinate system where the origin is the camera's perspective centre of the first frame. Thus, each depth frame was transformed into the 3D building model's coordinate system (Figure 6.21 (top right)) using its corresponding camera pose.

To assess the quality of the generated point cloud, CloudCompare's cloud-to-mesh (C2M) comparison tool was used to measure the distances between the collected point cloud and the 3D indoor model, i.e., the mesh (Figure 6.21 (top left)). The C2M function takes in a "reference" point cloud and a "compared" 3D mesh. A distance is assigned to every point in the reference cloud from itself, along its normal vector to the intersection of the mesh. Referring to Table 6.7, the C2M results revealed that 57.5% of the points collected were within 10 cm of the 3D model, 76.9% within 20 cm, 93.5% within 50 cm, and 95.7% within 1 m. The mean C2M distance for the points that were within 10 cm of the model was 6 cm ± 2 cm. The means for the points that were within 20 cm, 50 cm, and 1 m of the 3D model were 8 cm ± 4 cm, 8 cm ± 4 cm, and 13 cm ± 13 cm, respectively. These results were expected given the geometric accuracy of the model is 2 to 5 cm.

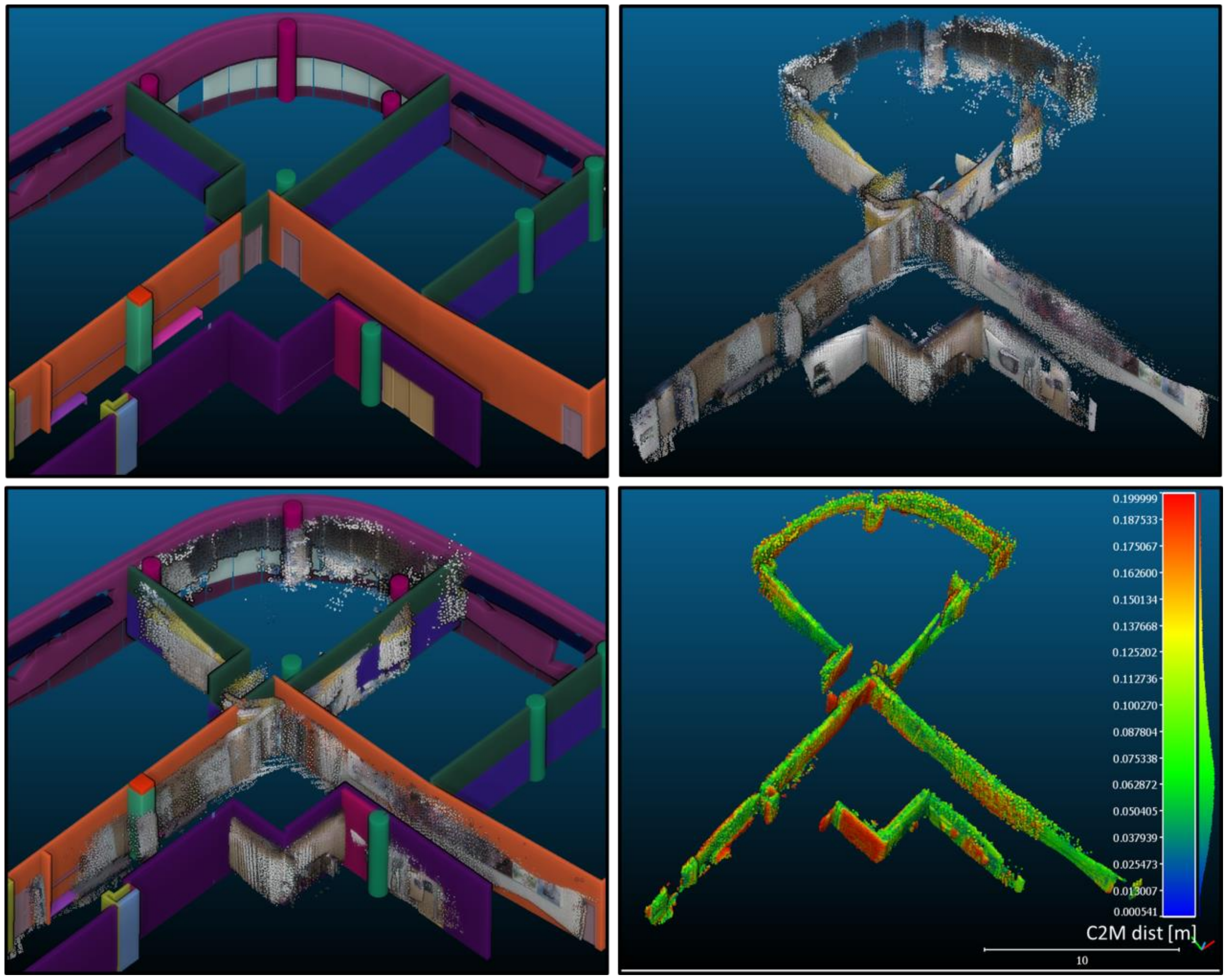


Figure 6.21. Top left) 3D Indoor model of the second floor of the study building. Top right) Point cloud captured using the integrated ViSP / RGB-D SLAM system. Bottom left) The collected point cloud overlaid on top of the 3D model. Bottom right) The cloud-to-mesh (C2M) distances (with 20 cm) between the collected point cloud and the 3D model.

Table 6.7. Statistics describing the quality of the generated point cloud. The total number of points was 809861. The second row shows the percentage of points less than the maximum C2M distance specified in the first row. The third and fourth rows show the mean and standard deviation, respectively, of the C2M distances for the points referred to in the second row.

Max. C2M distance [m]	**0.1**	**0.2**	**0.5**	**1.0**
% of points less than max. C2M distance [%]	57.5	76.9	93.5	95.7
Mean C2M dist. [m]	0.06	0.08	0.08	0.13
Std. Dev. C2M dist.[m]	0.02	0.04	0.04	0.13

6.7 Concluding remarks

In this chapter, a localization algorithm was proposed for UAV localization in GNSS-denied outdoor and indoor environments using only the onboard RGB camera. A model-based tracker capable of tracking large and complex objects in large environments provided a real time estimate of the relative pose between the camera and a georeferenced 3D model. In terms of accuracy and tracking maintenance, the results demonstrated a significant improvement in performance compared to the state-of-the-art. Specifically, the proposed solution improved 3D positioning precision in outdoor environments from 11.07 metres to 46 cm, and improved indoor 3D positioning precision from an unattainable result to 1.4 centimeters. The work in this section has contributed to Li-Chee-Ming and Armenakis (2013) and Li-Chee-Ming and Armenakis (2017a).

Future work may improve upon the weaknesses of the proposed model-based tracker, and its integration with a SLAM system. Specifically, the integrated system will fail when the model-based tracker and the SLAM system simultaneously fail to estimate the camera pose from an image. For instance, failure will occur when the SLAM system does not extract any features from an image, while none of the model is in the camera's field of view. A proposed solution is to predict the camera pose using a PVA (position-velocity-acceleration) dynamic model via a Kalman filter. Predicting the camera pose degrades the quality of the camera pose over time. Alternatively, complementary navigation sensors may be used to estimate the camera pose, for instance an IMU may provide pose estimates. However, as mentioned in Section 3.2, errors in IMU observations cause errors in position and orientation estimates to grow without bound.

Another limitation of the proposed method is the assumption that straight lines are present in the environment. This assumption restricts the use of the navigation system to structured urban outdoor environments and indoor scenes. Further, an assumption is made that vertical lines in the environment are vertical in the image, implying that images are horizontal to oblique, not nadir (downward facing). Expanding the use cases of the proposed solution would entail developing similar image-to-model matching techniques that use features present in unstructured (natural) environments, such as mountainous, forested, and rural areas. For example, Habib et al. (2000) match irregular linear features (natural lines) using a generalized Hough transform for the purposes of pose estimation. Geva et al. (2015) use a digital terrain model (DTM) in the pose estimation process from image sequences captured in nadir direction from a UAV flying in non-urban areas.

7 A MAP-BASED ABSOLUTE LOCALIZATION ALGORITHM

The previous work assumed an approximate camera pose was available to generate the synthetic image used in the matching process. The approximate pose may be obtained from the UAV's navigation system, or from a prediction from the previous epoch's camera position. A complementary algorithm is now proposed that does not make this assumption. That is, correspondence is established between an image captured by the UAV and the 3D model when approximate camera parameters are not available.

Lahdenoja et al. (2015) identified that this initialization and recovery phase is often done manually. However, when this process is automated via tracking-by-detection techniques, either fiducials (e.g., landmarks or markers) or the object's texture is used. The third contribution of this dissertation is a map-based absolute localization algorithm that is based on matching geometric features.

Quickly estimating the UAV's position and orientation is crucial, as the UAV travels rapidly. Fast pose estimation ultimately requires an efficient search of the 3D model (database). Geometric hashing (Wolfson and Rigoutsos, 1997) has been widely used to determine the feature correspondence between video frames and the 3D model of the environment in on-line applications. Geometric hashing is well-suited for this application as the extracted image features are not compared with every feature in the database; instead, the search space is strategically reduced such that only relevant information is accessed. Geometric hashing is a simple, efficient, and robust feature pattern matching method between two datasets. The main steps of the proposed approach are:

1) Vertical Line Features are extracted from the video frame.
2) Geometric hashing matches the extracted image features with their corresponding model features in the database.

3) The camera position and orientation parameters are estimated as a function of the matched feature locations in the map using a photogrammetric bundle adjustment solution.

Notably, the proposed approach estimated the initial approximations for the exterior orientation parameters required by model-based trackers. It was assumed that the onboard sensors (i.e., the GPS, IMU, and magnetometer) were not available to determine the UAV's pose. Further, an assumption was made that vertical lines in the environment remain approximately vertical in the image, implying that images were captured from a horizontal to oblique orientation, not nadir (downward facing).

7.1 Introduction to geometric hashing

Geometric hashing (Lamdan and Wolfson, 1988) is a map-based absolute localization technique. Geometric hashing is used in computer vision to match features against a database of such features. It is a highly efficient technique as matching is possible even when features have undergone geometric transformations or are partially occluded. Further, this method has low computational complexity as it is inherently parallel. Performing the search using multiple cooperating processors has been implemented (Lamdan et al., 1990; Grimson and Huttenlocher, 1988). Cham et al. (2010) demonstrated an increase in search speed, at the expense of accuracy, when geometric hashing is used in the matching stage of their 2D pose estimation framework.

Similar object recognition systems include the SCERPO vision system (Spatial Correspondence, Evidential Reasoning and Perceptual Organization) developed by Lowe (1987). The algorithm constructs perceptual groups of features (i.e., groups of lines that are parallel and in close proximity) and uses the viewpoint consistency constraint to recognize 3D objects from a single 2D image. The viewpoint consistency constraint states that a correct match is only found if the positions of the model's lines, back-projected according to the image's projective transformation, fit to the positions of the scene image lines. A Hausdorff-like distance (Huttenlocher et al., 1993) is often used to measure the goodness-of-fit between the image and projected model lines. Alternatively, the projected model is used to predict the locations of more image features, leading to the reliable verification or rejection of the initial match. Costa and Shapiro (2000) recognize 3D objects from 2D images by establishing correspondences of high-level features. High-level features are combinations of primitives such as lines, arcs, and ellipses. For matching, two of these high-level features are paired and characterized by their geometric relation (i.e., the two features share a common line or arc). These features are more distinct than primitive features, and more reliably matched. However, they are less likely to be completely detected in images because of measurement noise and occlusions. These features were stored in a hash table to speed up matching. The LEWIS system (Library Entry Working through an Indexing Sequence) developed by Rothwell et al. (1995) performed object recognition by matching invariants, which are features from an image of an object that remain constant regardless of the relative position and orientation of the object with respect to the image acquisition system. Similar to the high-level features, multiple primitive features were used to construct the invariants, thus matching was not be possible when invariants were not completely identified in the image. A hash table data structure was also used for on-line matching. The SoftPOSIT algorithm minimizes a global objective function to simultaneously establish correspondence between a 3D model and its perspective image and estimate the relative pose between the two (David et al., 2002). The POSIT (Pose from Orthography and Scaling with Iterations) technique (DeMenthon and Davis, 1995) was combined with *softassign*, an iterative correspondence assignment technique (Gold et al., 1998). The SoftPOSIT algorithm was later extended to use line features (David et al., 2003). Based on the performances of these object recognition systems, it is evident that the features used in matching should be reliably extracted and matched, as they depend on the data being processed (i.e., 2D camera image, 3D LIDAR point clouds, etc.) and the 3D model's data structure. Low computational time is also important as this is a real-time application.

The geometric hashing algorithm is divided into two stages: The pre-processing phase that generates the database, and the recognition phase that processes incoming images. The process' flow chart is provided in Figure 7.1 and the algorithm is summarized in the subsequent sections (Wolfson and Rigoutsos, 1997). Geometric hashing provides only the search engine portion of an object recognition system; the representation and extraction of the features are crucial inputs to geometric hashing. Deciding on the features to match depends on the available models and collected data.

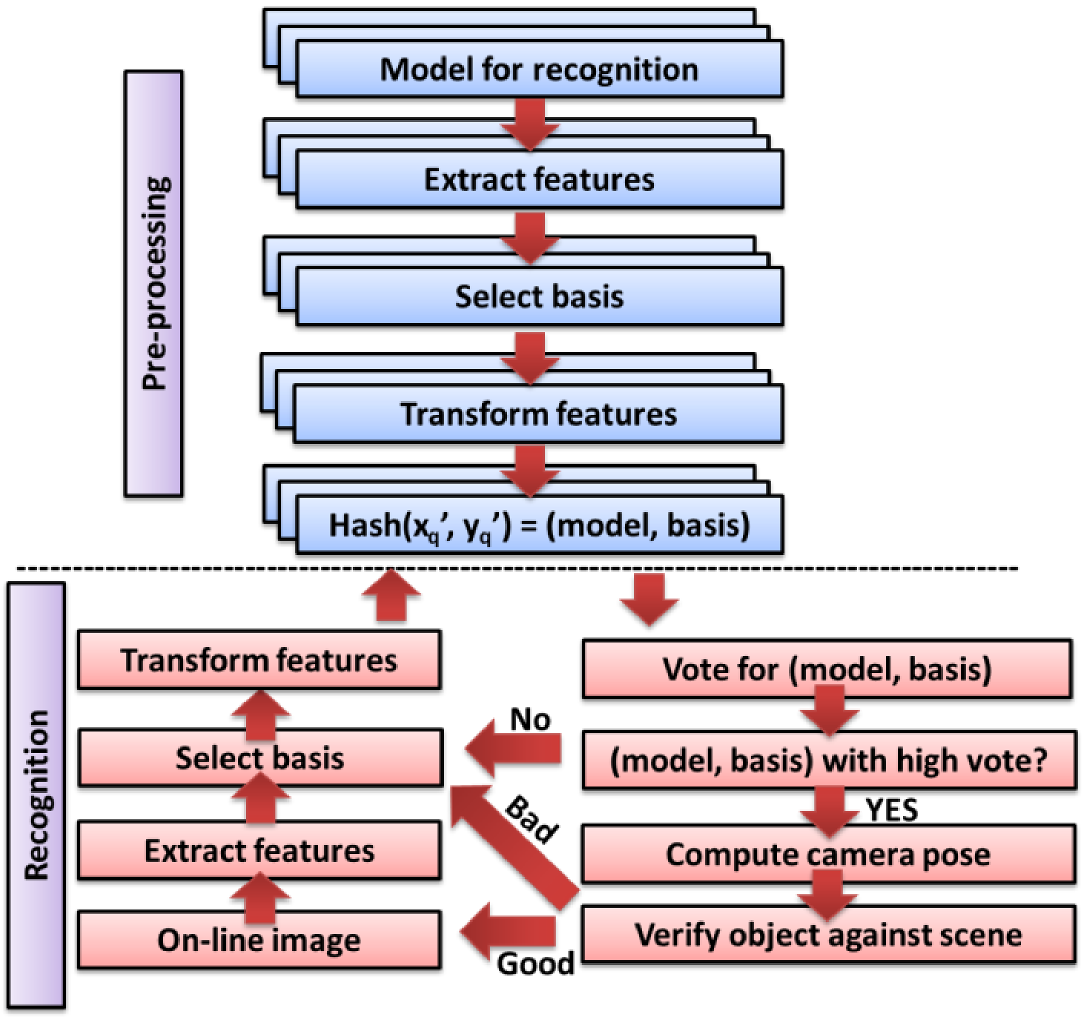


Figure 7.1. Flow chart for the geometric hashing algorithm, modified from (Lamdan and Wolfson, 1988).

7.1.1 The pre-processing phase

Conventional geometric hashing's pre-processing phase is described. For each model image, the following steps are performed:

1) Point features are extracted from the model image.

2) For each basis of 2 point features (assuming objects undergo similarity transformations), 3 point features (for affine transformation), or 4 point features (for projective transformation), the following steps are performed:
    a. The coordinates (x', y') of the remaining features are computed in a canonical reference frame defined by the basis.
    b. After proper quantization, the coordinates (x_q',y_q') are used as an index into a 2D hash table data structure and the information (M, basis), namely the model number and the basis used to determine (x_q',y_q'), are inserted into the corresponding hash table bin.

7.1.2 The recognition phase

When presented with an input image, the following steps are performed:

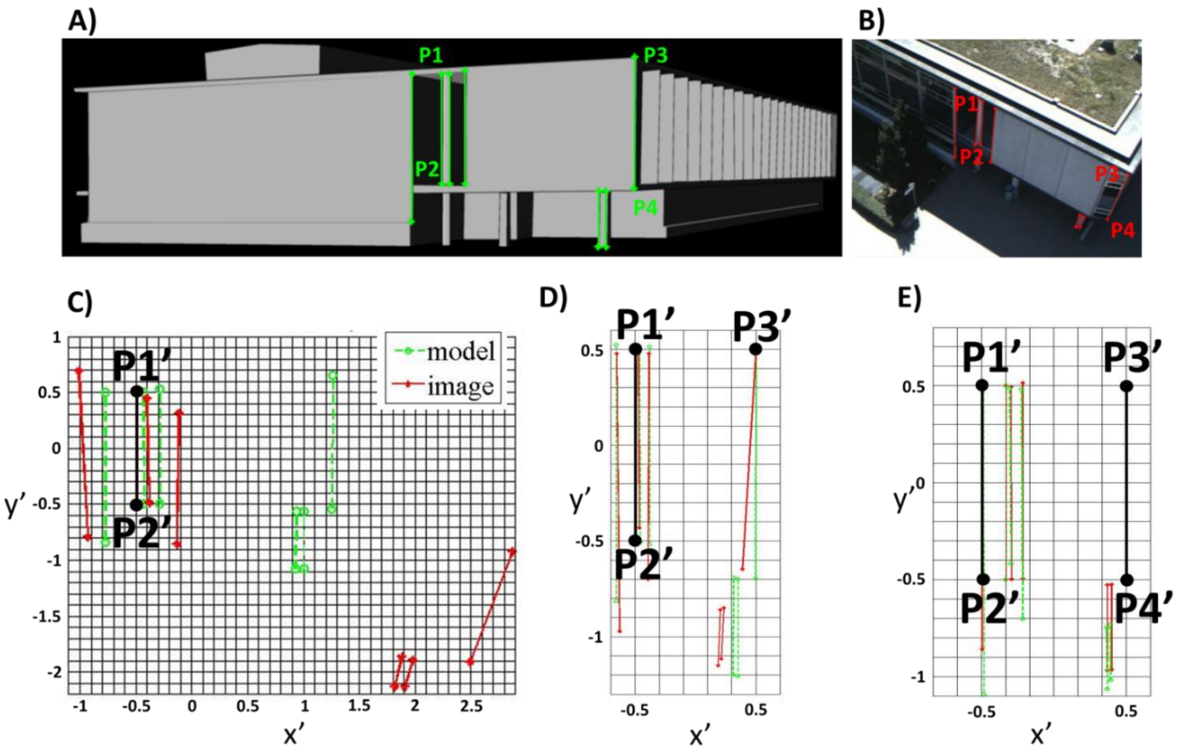
1) Point features are extracted from the image.
2) An arbitrary basis is chosen from the extracted features: 2 points when assuming objects undergo similarity transformation, 3 points for affine transformation, or 4 points for projective transformation.
3) The coordinates of the remaining feature points are computed in a canonical reference system *Oxy* that the selected basis defines.
4) Each such coordinate is suitably quantized and the appropriate hash table bin is accessed; for every entry found there, a vote is casted for the (model, basis) combination. Potential matches correspond to the (model, basis) combinations that receive more than a certain number of votes.
5) For each potential match, the transformation (i.e., the camera pose) that results in the best least-squares match between all corresponding feature pairs is recovered.
6) The features of the model are transformed according to the recovered transformation and verified against the input image features. If the verification fails, step 2 is repeated using a different image basis pair. If the verification passes, the next image is processed.

7.1.2.1 Selecting the geometric transformation

According to the geometric hashing algorithm, an input image will retrieve its corresponding model image from the database if a matching function finds a canonical coordinate system, *Oxy*, where corresponding model features and image features have the same positions, i.e., they fall into the same bin when they are transformed to *Oxy*. In order for the features to coincide, they must undergo a rotation, translation, and scaling, in two dimensions. In the simplest case, a 4 degree of freedom (DoF) similarity transformation may be used.

To estimate the four similarity transformation parameters for the model, two points belonging to the model are chosen to define a canonical frame of reference. For instance, the top and bottom corners of Vertical Line Features can be used to form the basis. For example, Figure 7.2A) shows Model M1. Figure 7.2C) shows the 2D hash table that is generated using Model M1's Vertical Line Feature 1, consisting of points **P1** (its top point) and **P2** (its bottom point), as the ordered basis. The Vertical Line Feature is scaled so that $\|\mathbf{P2'P1}\| = 1$ in the *Oxy* system. The midpoint between **P1** and **P2** is placed in *Oxy* in such a way that the vector $\overrightarrow{\mathbf{P2'P1'}}$ has the direction of the positive y axis. That is, the (x, y) image coordinate of **P1** and **P2** in the Model M1 are transformed to **P1'= (-0.5, 0.5)** and **P2'= (-0.5, -0.5)** in the *Oxy* system, respectively. These two corresponding points are used to solve for the similarity transformation parameters. The remaining Vertical Line Features of Model M1 (green lines in Figure 7.2A), to be used as support, are transformed to Oxy via the computed transformation parameters. In the quantized hash table, a record is kept in each of the bins where the remaining points land, noting that model M1 with Vertical Line Feature 1, expressed as (M1, 1), yields an entry in this bin.

Due to occlusions, it is not guaranteed that both model points **P1** and **P2** will appear in an image where model M1 is present during the recognition phase. Consequently, the model's features are encoded in all possible ordered basis pairs. Namely, the hash table contains entries of the form (M1, 1), entries of the form (M1, 2), and so on. The same process is repeated for the remaining models in the database (i.e., M2, M3, M4, etc.). Each hash table bin has a list of entries of the form (model, basis). Some hash table bins may receive more than one entry. As a result, each bin will contain a list of entries of the form (model, basis).



$$x' = ax - by + c$$
$$y' = bx + ay + d$$

$$x' = ax + by + c$$
$$y' = dx + ey + f$$

$$x' = \frac{ax + by + c}{gx + hy + 1}$$
$$y' = \frac{dx + ey + f}{gx + hy + 1}$$

Figure 7.2. A) Model M1 with extracted Vertical Line Features (green lines). B) Incoming image with extracted Vertical Line Features (red lines). Establishing a canonical reference frame using C) a two-point basis (**P1** and **P2**) and the similarity transformation, D) a three-point basis (**P1**, **P2** and **P3**) and the affine transformation, and E) a four-point basis (**P1**, **P2**, **P3**, and **P4**) and the projective transformation.

Geometric hashing can be applied to many other transformations. The main difference is the number of points used to define the basis for the reference frame. Specifically:

- 2-DoF translation uses a one-point basis, mapping model point **P1**(x, y) to **P1'** (0, 0).
- 3-DoF translation and rotation uses a two-point basis, mapping model point **P1**(x, y) to **P1'**(-0.5, 0.5), and **P2**(x, y) to **P2'**(-0.5, -0.5).

- 4-DoF similarity transformation uses a two-point basis, mapping model point **P1**(x, y) to **P1'**(-0.5, 0.5), and **P2**(x, y) to **P2'**(-0.5, -0.5).
- 6-DoF affine transformation uses a three-point basis, mapping model point **P1**(x, y) to **P1'**(-0.5, 0.5), **P2**(x, y) to **P2'**(-0.5, -0.5), and **P3**(x, y) to **P3'**(0.5, 0.5).
- 8-DoF projective transformation uses a four-point basis, mapping model point **P1**(x, y) to **P1'**(-0.5, 0.5), **P2**(x, y) to **P2'**(-0.5, -0.5), **P3**(x, y) to **P3'**(0.5, 0.5), and **P4** to **P4'**(0.5, -0.5).

Notably, increasing the number of points that form the basis significantly increases the computational complexity of the algorithm and decreases the probability of finding a match. Figure 7.2 reveals that the matching accuracy increases if a projective transformation is assumed, as opposed to a similarity or affine transformation, as the image features coincided the closest with the model features. This increase in matching accuracy is expected as objects undergo a projective transformation when projected to a camera's image plane. Using an affine transformation is also a viable choice as fewer points are needed to define the basis, which increases the probability that all of the basis points extracted from the image will be present in the model. Further, the affine transformation parameters can be more accurate as less error is propagated from the image measurements of the basis points. Many investigators suggest that the nonlinear perspective projection can be accurately substituted with a linear affine approximation when the relative depth of object features is small compared to the distance of the object from the camera (David et al., 2002).

7.2 A map-based absolute localization solution for outdoor environments

The proposed strategy to establishing correspondence between the outdoor 3D building model and the UAV's video imagery is to generate a database of images of the building model, referred to in this work as synthetic images, captured from various vantage points. In the recognition phase, features are extracted from the current UAV video frame and matched with the model features extracted from the synthetic images. That is, based on the image from the UAV video camera, georeferenced buildings that appear in the image are retrieved from the 3D database and then the retrieved buildings are used as ground control to compute the camera pose. The main challenge, and focus of this work, is retrieving the correct building from the 3D database, which is essentially matching features extracted from an image with features in a database. This process is divided into two components: 1) the pre-processing phase generates the database, and 2) the recognition phase processes incoming images.

7.2.1 The pre-processing phase

In a pre-processing stage, a database of the Vertical Line Features extracted from the 3D building models was created to efficiently recognize Vertical Line Features extracted from UAV's images.

7.2.1.1 Synthetic images of the 3D model

To generate the database of synthetic images, 3D building models were loaded into Google Sketchup, a 3D computer graphics software product (www.sketchup.com/). For each building model, such as the Lassonde Building Model shown in Figure 7.3A), the normal vectors $\boldsymbol{\eta}$ of the building (TIN) faces were averaged, to $\boldsymbol{\eta}_{ave}$, to determine the prominent directions that building walls were facing. Each prominent normal direction represented a group and each face was then assigned to the group whose average normal was closest to its normal. For example, the normals of the faces of the Lassonde Building were separated into four prominent normal directions: North ($0^o \leq \boldsymbol{\eta}_{ave} < 90^o$), East ($90^o \leq \boldsymbol{\eta}_{ave} < 180^o$), South ($180^o \leq \boldsymbol{\eta}_{ave} < 270^o$), or West ($270^o \leq \boldsymbol{\eta}_{ave} < 360^o$) facing (Figure 7.3B).

A synthetic image was generated for each group. This involved setting Sketchup's camera pose such that all of the TIN faces of a specific group were 1) inside the camera's field of view, and 2) facing the camera. Sketchup defines the camera orientation via a Target vector and an Up vector. The Target, a point in space that the camera's optical axis intersects, is used to define the pitch and yaw of the camera. The Up vector, the direction that the top of the camera is facing, is used to define the roll of the camera.

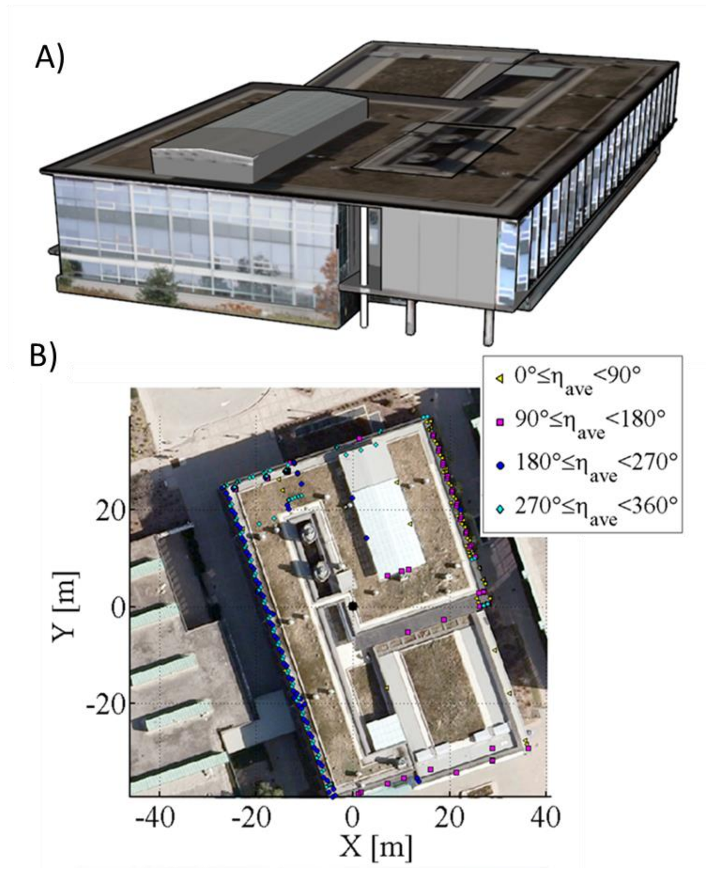


Figure 7.3. A) The Lassonde Building model. B) The Lassonde Building model's normal vectors classified into four prominent groups.

A process was developed to determine the camera pose that allowed all of the group's faces to appear in the camera's field of view. The steps for setting the camera pose for one group were as follows:

1) The camera's target was set to the centroid of all the faces in the group. The Up vector was made parallel to the vertical axis (signifying a zero degree roll angle).
2) The camera position started at the Target position (i.e., the centroid of all the faces in the group) and moved away from the Target, in the direction of its optical axis (i.e., pointing towards the Target). An efficient point-in-polygon strategy (Haines, 1994) was used to detect when all faces of the group fell inside the camera's field of view (FOV), as it is depicted in Figure 7.4A). An image was captured once this condition was met. Figure 7.4B) shows the synthetic images captured for all four groups of the Lassonde Building.

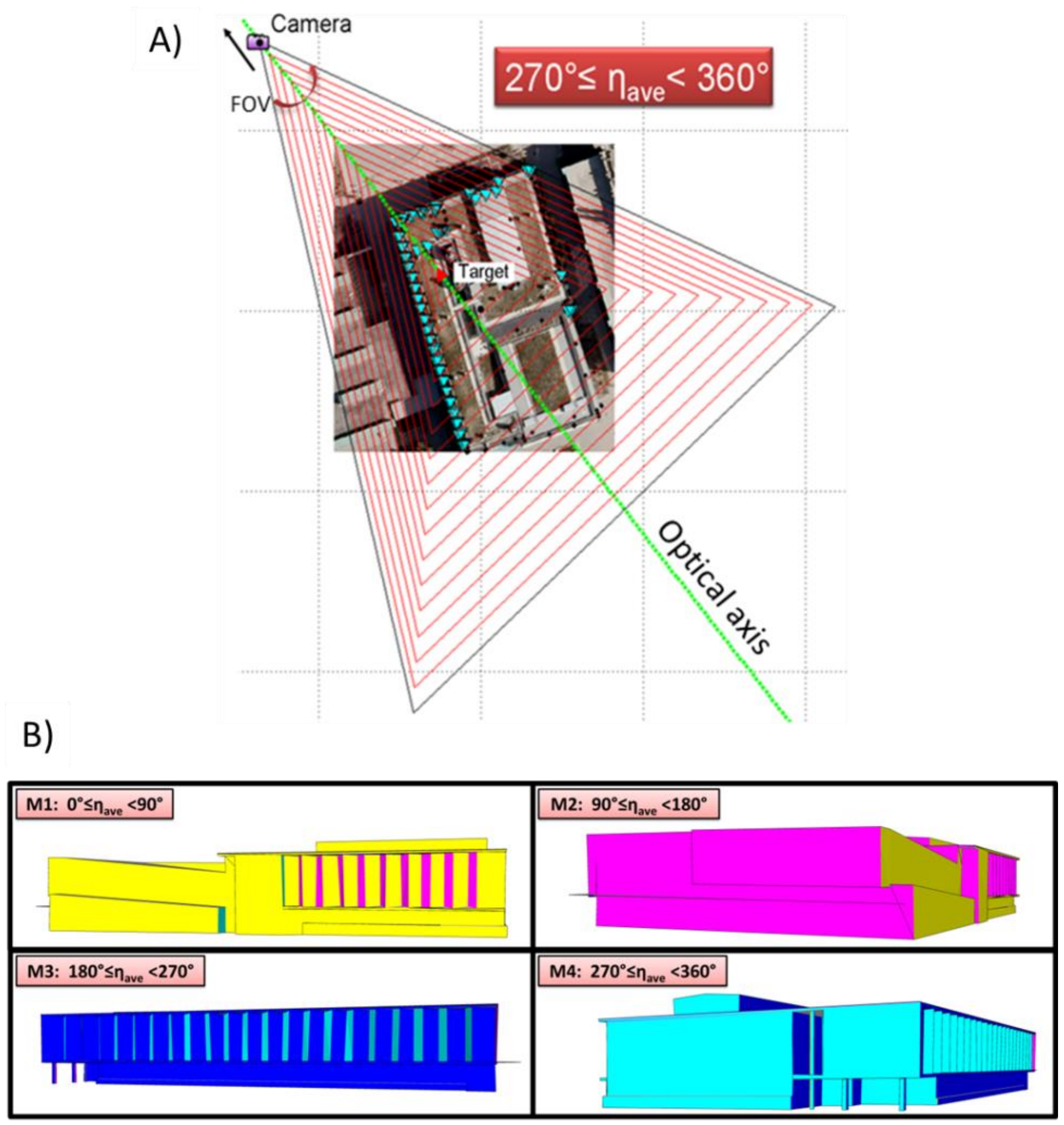


Figure 7.4. A) Setting the virtual camera's position and orientation. B) Four synthetic images for the Lassonde Building model.

7.2.1.2 Extracting model features

Using the same method described in Section 6.3, Vertical Line Features were similarly extracted from the synthetic images. The model's vertices were classified according to corner type, and vertical edges were classified as either Left or Right Vertical Lines. For example, Figure 7.5A) shows model M1 with the extracted Left and Right Vertical Line Features, numbered 1 through 27. Similarly, Figure 7.5B) shows model M4 with the extracted Left and Right Vertical Line

Features, numbered 1 through 36. The information was encoded and stored into a hash table. If an image containing this information was collected, it could be matched with the model using this table. The following section describes this process.

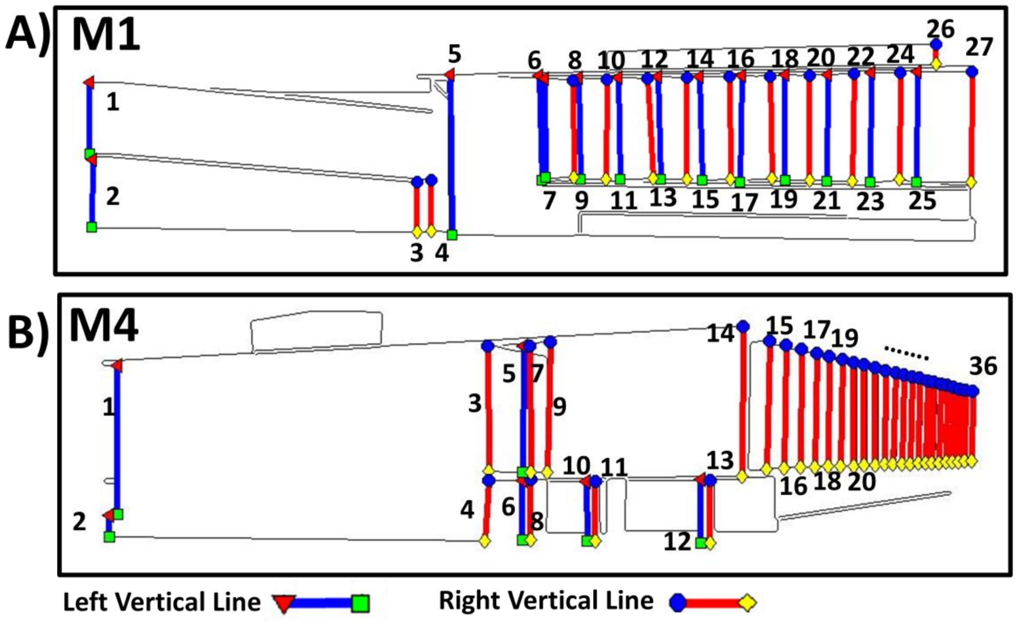


Figure 7.5. A) Model M1 Vertical Line Features 1 to 27, B) Model M4 Vertical Line Features 1 to 36.

7.2.1.3 Generating the database of hash tables

The geometric hashing algorithm was used to generate the database of hash tables. For example, Figure 7.6 shows two hash tables: A) Model M4 using the endpoints of Vertical Lines Features 1 and 14 (from Figure 7.5A) expressed as the basis, or (M4, (1, 14)), explicitly: **P1**(x, y) is the Top Left Corner of the Left Vertical Line 1, which mapped to **P1**'(-0.5, 0.5), **P2**(x, y) is the Bottom Left Corner of the Left Vertical Line 1, which mapped to **P2**'(-0.5, -0.5), **P3**(x, y) is the Top Right Corner of the Right Vertical Line 14, which mapped to **P3**'(0.5, 0.5), and **P4**(x, y) is the Bottom Right Corner of the Right Vertical Line 14, which mapped to **P4'**(0.5, -0.5). Similarly, Figure 7.6B) shows the hash table for (M4, (14, 28)).

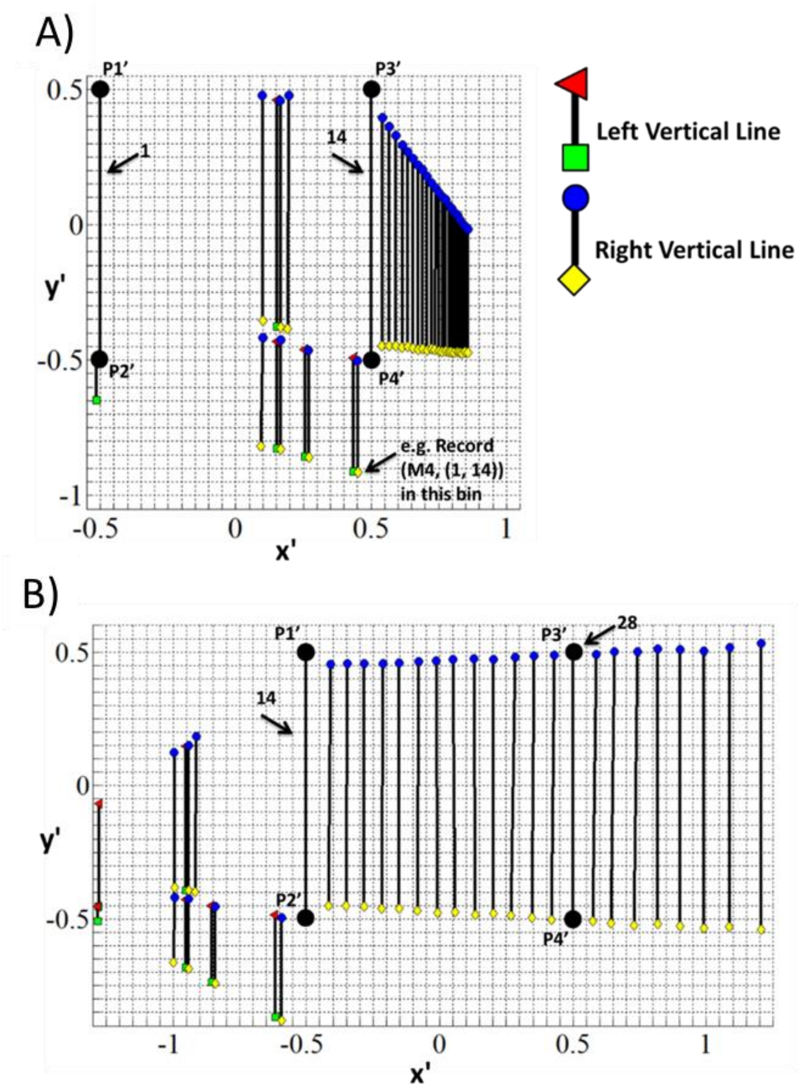


Figure 7.6. Hash tables for A) Model M4 with Vertical Line Features 1 and 14 as the basis, i.e., (M4, (1, 14)), and B) Model M4 with Vertical Line Features 14 and 28 as the basis, (M4, (14, 28)).

7.2.2 The recognition phase: Image-to-model registration

In the recognition phase, features were extracted from the incoming video image and matched with features in the database. As the projective transformation was used in this work, the top and bottom corners of two Vertical Line Features extracted from an image were chosen as a candidate basis. As in Section 7.2.1.3, this basis defined a canonical coordinate system *Oxy*, where the image points **P1** to **P4** mapped to the points **P1'** to **P4'** in *Oxy*. The coordinates of all the other points acted as support as they were transformed to *Oxy*. The corner points were then mapped to the hash table and all the entries in the corresponding hash table bin received a vote. Models accumulating high counts had high probability to be present in the scene. The computational complexity of this process is linear in the number of features.

The search space was reduced by separating Vertical Line Features into Left Vertical Line Features and Right Vertical Line Features. That is, a Left Vertical Line Feature extracted from an image only searched for the Left Vertical Line Features in the database. If two Vertical Line Features extracted from an image corresponded to a basis on one of the models, all of the unobstructed points belonging to this model accumulated a vote. If there was a sufficient number of votes for a (model, basis) combination, a subsequent test verified a correct match. Notably, recognition was still possible if the object was partially occluded in the image, as long as a sufficient number of points hashed to the correct bins.

Since every basis combination was redundantly stored in the database, in theory, the first pair of Vertical Line Features arbitrarily chosen from the image as a basis that matched with a (model, basis) combination in the database should have resulted in a successful search. However, experimentation revealed that the accuracy of the matches increased if the two Vertical Line Features chosen from the image as a basis, were spread across the image, with the supporting features located in between the two Vertical Line Features chosen as a basis. Further, the two Vertical Line Features chosen as a basis should be relatively long compared to the other features because transforming the remaining features in these conditions becomes analogous to an interpolation, as opposed to a less accurate extrapolation. As a result of this finding, the video images were stitched together to generate a single image of the entire building façade in real-time using OpenCV's implementation of (Brown and Lowe, 2007). A larger field of view allowed for an increased number of features to be present in the image, and increased in the maximum separation between the Vertical Line Features. David and Ho (2011) demonstrated that onmidirectional images also provide a wide field of view in this application. Further, selecting two Vertical Line features as a basis incorporated a threshold for the separation between to Vertical Line Features, along with a threshold for the length of the lines. Figure 7.7A) shows two images stitched together, Figure 7.7B) shows the Canny edge image and the extracted Vertical Line Features, and Figure 7.7C) shows the hash table which received the most votes, only the corresponding features are plotted. Section 0 provides experimental results demonstrating the proposed absolute map-based localization technique successfully estimating the UAV's initial pose estimate in outdoor environments.

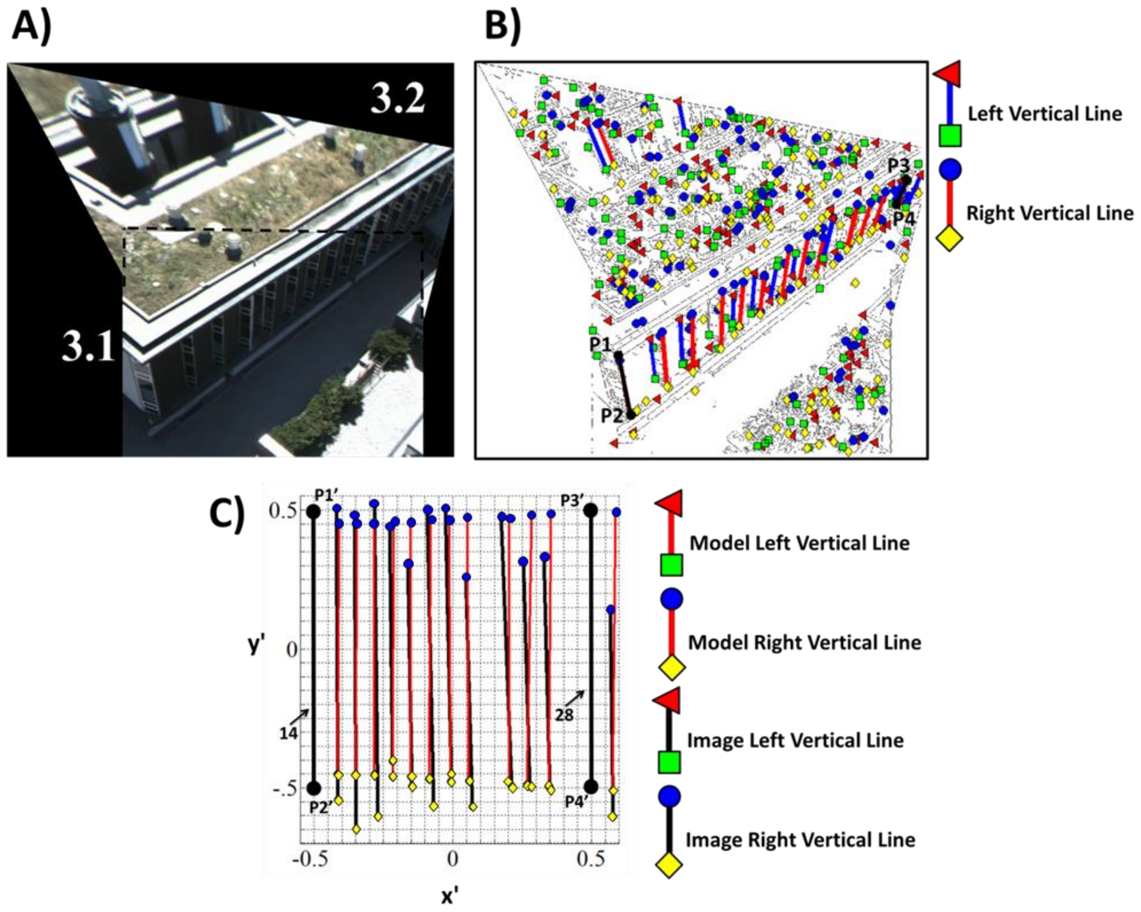


Figure 7.7. A) Two images stitched together, B) the Canny edge image and the extracted Vertical Line Features, C) the hash table (M4, (14, 28)), which received the most votes, only the corresponding features are plotted.

7.3 A map-based absolute localization solution for indoor environments

The proposed strategy to finding correspondence between the 3D indoor model and an image, without a prior estimate of the camera pose, was as follows:

1) In the pre-processing phase, a database was generated consisting features extracted from panoramic images of the 3D indoor model, referred to in this work as synthetic panoramic images, captured from various vantage points in the model.
2) In the model-based tracker's initialization phase:
    a. A set of images was captured by rotating an RGB camera 360 degrees about its vertical axis.
    b. A panoramic image was generated (e.g., using OpenCV).
    c. Features were extracted from the panoramic image and matched with the model features in the database of synthetic panoramic images using geometric hashing.
    d. The camera pose was computed through photogrammetric space resection using the retrieved model features as ground control.

It was decided to extract features from panoramic images instead of individual images because geometric hashing searches the database for a group of features extracted from an image; a larger field of view allows for each group to contain a larger number of features, increasing the distinctiveness of the group of features, and in turn increases the probability of retrieving the correct model from the database.

The following section explains the choice of features as it depended on the available models and collected data. Subsequent sections propose an improvement to the geometric hashing algorithm, and the modifications made to suit the specific application of navigation in GPS-denied indoor environments.

7.3.1 Feature selection

Linear features were again chosen to establish correspondence between the 3D model and the image. The chosen linear feature, referred to in this work as the Vertical Line Feature, consists of two end points connected by a vertical line. In agreement with Cham et al. (2010), Vertical Line Features were chosen because vertical lines commonly appear in structured environments. Further, vertical lines in the environment remain vertical in a cylindrical panoramic image, while horizontal lines in the environment become curved.

Estimating the 8 projective transformation parameters requires four 2D points; this implies that image-to-model matching will succeed if the top and bottom points of two Vertical Line Features extracted from the image are present in the 3D model. Section 7.2.2 suggested that improved performance is achieved if the two Vertical Line Features chosen from the image as a basis are spread across the image, with the supporting features located in between the two Vertical Line Features chosen as a basis. Further, the two Vertical Line Features chosen as a basis should be relatively long compared to the other features. This behaviour is expected, as transforming the supporting features in these conditions becomes analogous to an interpolation, as opposed to a less accurate extrapolation. As a result of this finding, it was decided to extract both model and image Vertical Line Features from 360 degree panoramic images. A larger field of view allows for an increased number of features to be present in the image, and increased the maximum separation between the Vertical Line Features used as a basis.

7.3.2 Synthetic panoramic images of the 3D model

To generate the database of synthetic panoramic images, 3D indoor building models were loaded into the Gazebo Robot Platform Simulator (Open Source Robotics Foundation, 2017), which is a 3D simulator with a physics engine. Gazebo is capable of simulating robots and a variety of sensors in complex and realistic indoor and outdoor environments.

For each indoor model, such as the Bergeron Centre building shown in Figure 7.8, an equally spaced grid of simulated cameras was spread across the extent of the model. At each grid location, the camera rotated 360 degrees at 60 degree intervals and captured a synthetic image at each interval (Figure 7.9A). The known poses of each image were used to automatically generate cylindrical panoramic images for each grid location (Figure 7.9C)). Vertical Line Features were extracted from each synthetic panoramic image and stored in a hash table. Figure 7.9 shows a synthetic panoramic image captured in Room B1 of the Bergeron Centre. The figure also shows the extracted Vertical Line Features.

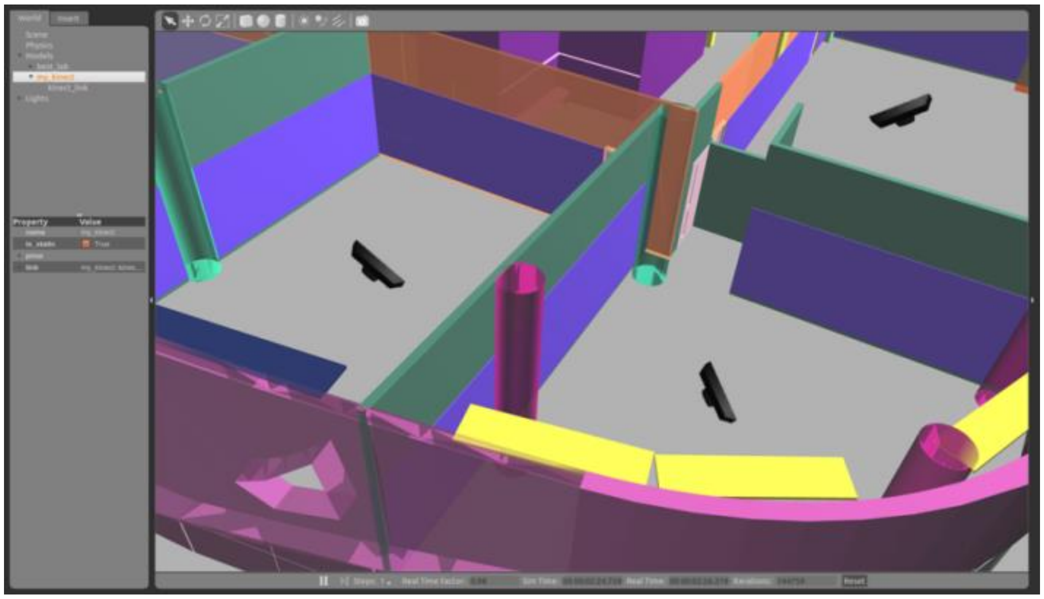


Figure 7.8. Equally spaced grid of simulated cameras capturing 360° synthetic panoramic images in the Bergeron Centre 3D CAD model.

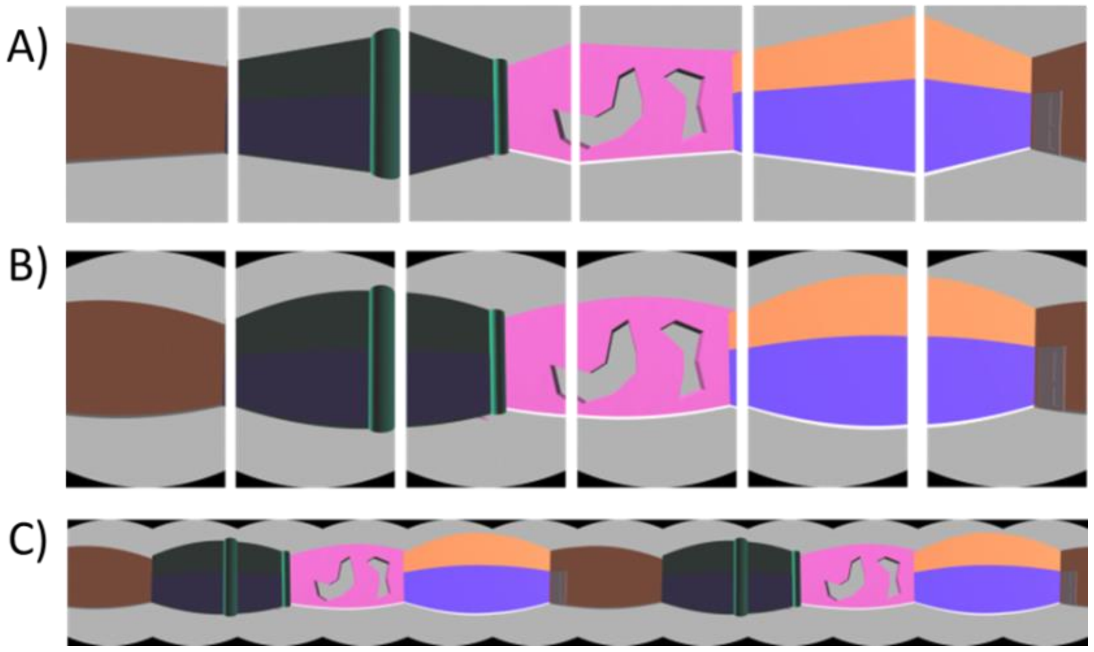


Figure 7.9. Generating a synthetic panoramic image: A) Synthetic images captured by the virtual camera at one grid location. B) Cylindrical projection of synthetic images (the radius of the cylinder is equal to the focal length of the camera). C) Resulting synthetic panoramic image.

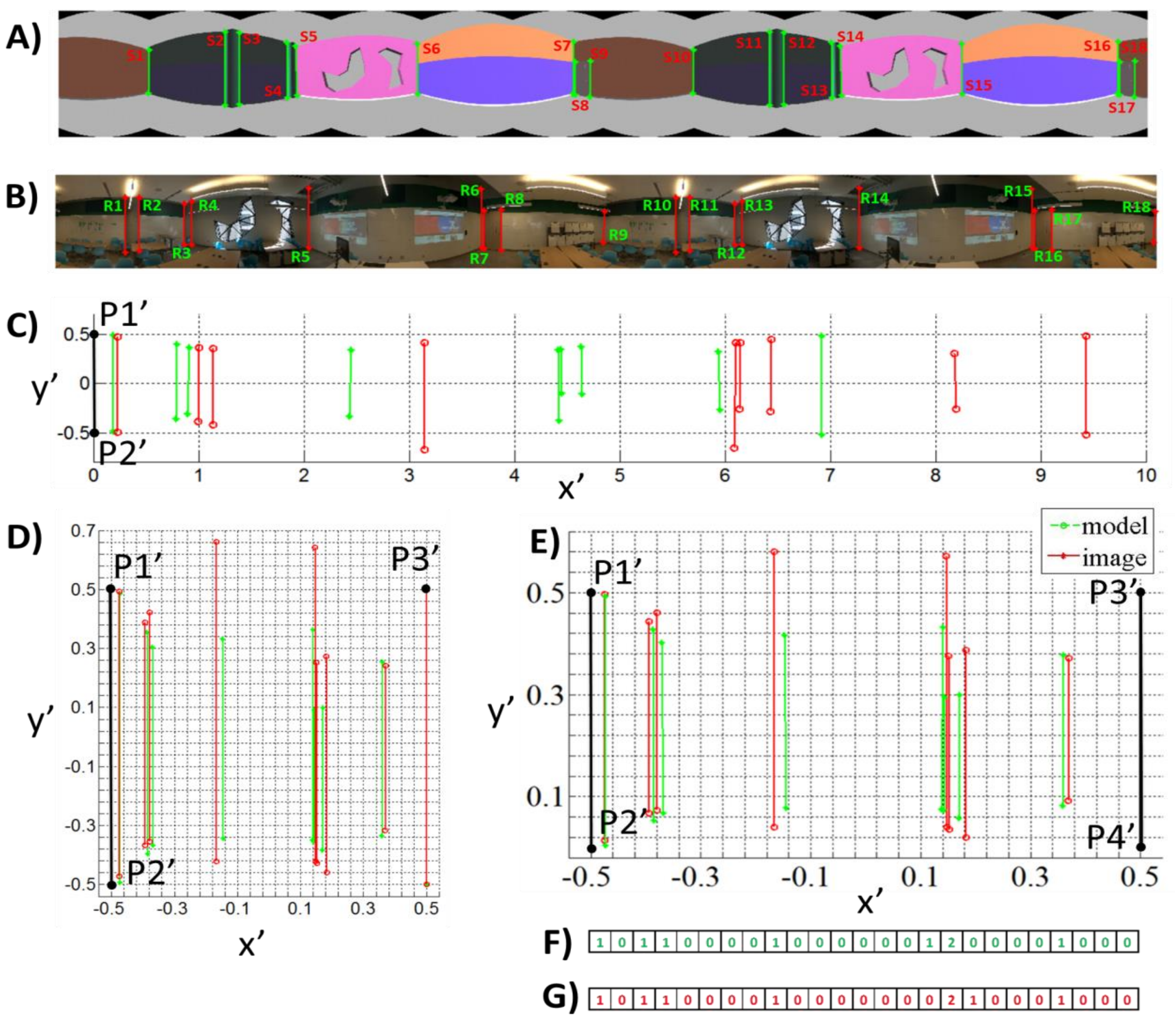


Figure 7.10. A) Synthetic panoramic image of Room B1 with extracted Vertical Line Features. B) Incoming panoramic image of Room B1 with extracted Vertical Line Features. Establishing a canonical reference frame using C) similarity transformation: 2D hash table populated using as 2-point bases Vertical Line S2 from the synthetic panoramic image and Vertical Line R1 from the incoming panoramic image, D) affine transformation: 2D hash table populated using as 3-point bases Vertical Line S2 (and the top corner from its duplicate Line S11) from the synthetic panoramic image and Vertical Line R1 (and the top point from its duplicate Line R10) from the incoming panoramic image, and E) projective transformation: 2D hash table populated using as 4-point bases Vertical Line S2 (and its duplicate Line S11) from the synthetic panoramic image and Vertical Line R1 (and its duplicate Line R10) from the incoming panoramic image. F) Series generated from the Vertical Line Features of the model and G) the incoming image. Each element of the series is the number of Vertical Lines feature in the 1D hash table's bin.

7.3.3 Improving geometric hashing

This section proposes an improvement of the geometric hashing algorithm. The probability of finding a correct match in the database is increased while decreasing the size of the database. The incoming panoramic and the synthetic panoramic images were captured such that they began and ended at the same pose, i.e., the horizontal field of view is exactly 360 degrees, allowing a panoramic image to be concatenated with itself. Consequently, each Vertical Line Feature appeared twice, separated by all of the other Vertical Line Features, in the panoramic image. Thus, instead of the four point basis consisting of two different Vertical Line Features, only one Vertical Line Feature was used to estimate the projective transformation parameters. That is, the top and bottom point of a Vertical Line Feature mapped to two points of the basis, the other two points were mapped to the duplicated Vertical Line Feature belonging to the concatenated panoramic image. Only the lines in between these two Vertical Line Features were projected to the hash table and used as support. The probability of finding a match is increased in this case as only one Vertical Line Feature needs to be matched instead of two. For example, Figure 7.10A) shows a synthetic panoramic image of room B1, Vertical Line Feature S2 and its duplicate S11 are used as a basis for the model data. All of the model lines in between these two lines (i.e., Lines S3 to S10) are projected into the canonical coordinate system (green lines in Figure 7.10C)). Similarly, Figure 7.10B) shows a panoramic captured of Room B1, Line R1 and its duplicate Line R10 are used as a basis for the input image data. All of the lines in between the two basis lines (i.e., Lines R2 to R9) are projected into the canonical coordinate system (red lines in Figure 7.10C)). It is evident in Figure 7.10C) that the image features and corresponding model features do not exactly coincide because of error from the model, the quantization error of the hash table, error in the image measurements, and occlusions. The size of the bin is a critical parameter. If the bins are too large, then there will be too many false positives. If the bins are too small, then there is insufficient provision for noise in the input. Results from using the conventional geometric hashing voting approach (Lamdan and Wolfson, 1988) revealed that using the corner points of Vertical Line Features produced many false candidates in the recognition phase because the required search area was large. However, the majority of this error was in the vertical axis because the model's corner points were often not visible in the real image as they were occluded by clutter in the environment.

In an effort to reduce the number of model candidates returned from the voting stage, a different voting strategy was developed that leveraged the use of vertical lines instead of individual points: A smaller search area extended only in the horizontal direction. That is, a (model, basis) combination received a vote if the midpoint of an image line was within a certain horizontal distance from one of its supporting model lines' midpoints. Experimentation revealed that, for this dataset, the average horizontal error separating corresponding image and model lines in the canonical coordinate system was 0.04±0.02 units. Thus, the bin size was chosen to be 0.04 units. Notably, this strategy reduced the two dimensional search space to 1 dimension (e.g., a 25 bin hash table as shown in Figure 7.10D) for the model data and Figure 7.10E) for the image data. This significantly reduced the size of the database and the search time.

7.3.3.1 The similarity function

A score function was introduced to measure the similarity between the supporting model data and the supporting image data. Geometric hashing's similarity score is based simply on counting the number of votes, which is not sufficient because it depends only on the number of supporting lines. In other words, the probability of an image matching a certain model increases as the number of (image and model) features increases because the probability of an image feature projecting into a model feature's bin increases. A standardized scoring method was required, one that did not depend on the number of features in the image or model. The chosen similarity score was the normalized cross-correlation (NCC) (Lewis, 1995). Specifically, a series (e.g., 25 numbers long for a bin size of 0.04) was generated from the hash table. Each element of the series contained the number of lines in the bin. For example, Figure 7.10D) shows the series generated from the model's supporting data in green. Figure 7.10E) shows the series generated from the image's supporting data in red.

If corresponding image and model lines are chosen as a basis, the resulting NCC coefficient of the two series will be high in an absolute sense, e.g., an NCC coefficient higher than 0.8. Further, corresponding image and model lines produce high NCC coefficient in a relative sense. For example, Table 1 shows the NCC coefficient resulting from using the row's model line and the column's image line. For instance, Model Line S2 from Figure 7.10A) and Image Line R1 from Figure 7.10B) are corresponding lines. The series they generate are provided in Figure 7.10 D) and E), respectively. The NCC coefficient of these two series is 0.86, this value is found in its corresponding row and column in Table 7.1. Two conditions are met here: 1) This NCC coefficient is the highest in its row and 2) this NCC coefficient is the highest in its column. Having the highest NCC coefficient in its row implies that for Model Line S2, the best matching image line in this particular panoramic image is Line R1. Similarly, having the highest NCC coefficient in its column implies that for Image Line R1, the best matching model line in this particular synthetic panoramic image is Model Line S2. An image-model line pair could be considered candidate matches if one or, more stringently, both of these conditions are met. To further increase the stringency, one could compare the highest NCC coefficient against the second highest NCC coefficient in a row/column. There is more confidence in an image-model line match if its NCC coefficient is much higher than the NCC coefficient of any other match in its row and column, for example 20% higher.

Table 7.1. Table of NCC coefficients used to identify corresponding Vertical Line Features from the input panoramic image and synthetic panoramic image captured in Room B1.

	R9	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**
S1	**0.91**	0.85	0.54	0.68	0.58	0.62	0.48	0.57	0.52
S2	0.83	**0.86**	0.58	0.72	0.62	0.64	0.52	0.59	0.56
S3	0.61	0.75	**0.87**	0.86	0.61	0.70	0.60	0.42	0.67
S4	0.54	0.65	0.66	**0.90**	0.66	0.71	0.80	0.51	0.66
S5	0.74	0.53	0.55	0.69	**0.91**	0.92	0.68	0.35	0.70
S6	0.67	0.58	0.49	0.69	0.81	**0.92**	0.81	0.45	0.60
S7	0.61	0.46	0.54	0.70	0.60	0.58	**0.89**	0.63	0.48
S8	0.55	0.61	0.54	0.61	0.55	0.41	0.77	**0.79**	0.49
S9	0.49	0.55	0.75	0.66	0.52	0.65	0.48	0.47	**0.89**

7.4 Results: A novel map-based absolute localization algorithm

The following experiments apply the proposed model-based absolute localization solution to estimate the UAV's camera pose with no prior pose information. Experiments were performed in both outdoor and indoor environments. The resulting pose was used to initialize the map-based incremental localization algorithms described in Chapters 5 and 6.

7.4.1 Experiment 1: A map-based absolute localization solution for outdoor environments

The images collected by the UAV that were presented in Section 5.5 are processed here using the proposed map-based absolute localization system for outdoor environments. The imagery was downsampled to 1 megapixel to enable real-time image processing. The 3D virtual building model of York University's Keele Campus (Armenakis and Sohn, 2009) was used as the known 3D map environment. This was the same model used in the experiments presented in Section 5.5. This 3D CAD model served two purposes in the proposed approach. Firstly, it provided the necessary level of detail of linear features (vertical lines) for feature matching. Secondly, it provided ground control points to achieve photogrammetrically sub-metre accuracies of the positional elements of the exterior orientation.

Referring to Table 7.2, a threshold value of one quarter (¼) of the image width was used for the line length and line separation for the outdoor dataset. This threshold was chosen because it reduced the number of candidate lines to an adequate amount for real-time matching onboard the UAV, while yielding an adequate number of support lines in between the two basis lines to reliably retrieve the correct synthetic image from the database. In the recognition phase, an image retrieved a candidate model image from the database if more than 70% of the Vertical Line Features in the image were found in the model image.

Table 7.2: Map-based absolute localization threshold values used in the outdoor experiment.

Threshold parameter	**Threshold value**
Line length	¼ image width
Vertical line separation	¼ image width
Number of votes	70%

It is evident in Figure 7.11, Figure 7.12, and Figure 7.13 that the image features and corresponding model features do not exactly coincide in their canonical coordinate system due to errors from the model, the quantization errors of the hash table, errors in the image measurements, and occlusions. The size of the bin is a critical parameter. If the bins are too large, then there will be too many false positives. If the bins are too small, as they are in Figure 7.11 to Figure 7.13, then there is insufficient provision for noise in the input. (Gavrilla and Groen, 1992) use small bins, but compose a circular pattern of multiple bins for each bin entry. That is, to accommodate for the error in the recognition phase when a point (x, y) hashes to a location (u, v), the vote count is increased for all the entries in the hash table that lie near the point (u, v). Each vote is given a weight that decreases as the distance between (u, v) and (x, y) increases. While a simplified solution involves assessing a circular or rectangular region of the table instead of a single bin (Lamdan and Wolfson, 1989), a Bayesian maximum-likelihood approach is a common alternative used to more accurately estimate the size, shape, and orientation of the neighbourhood, and in turn determine the voting weights (Rigoutsos and Hummel, 1995).

After all scene points were processed, the vote for a particular model/basis combination was determined. When all weighted votes were tallied, the model/basis combinations whose total accumulated evidence exceeded a threshold were passed onto a verification process. As mentioned above in Table 7.2, a threshold of 70% of expected votes was used in the experiments. If the maximum accumulated evidence for no model/basis combination exceeded the threshold, the algorithm continued by iterating through different candidate basis pairs from the image.

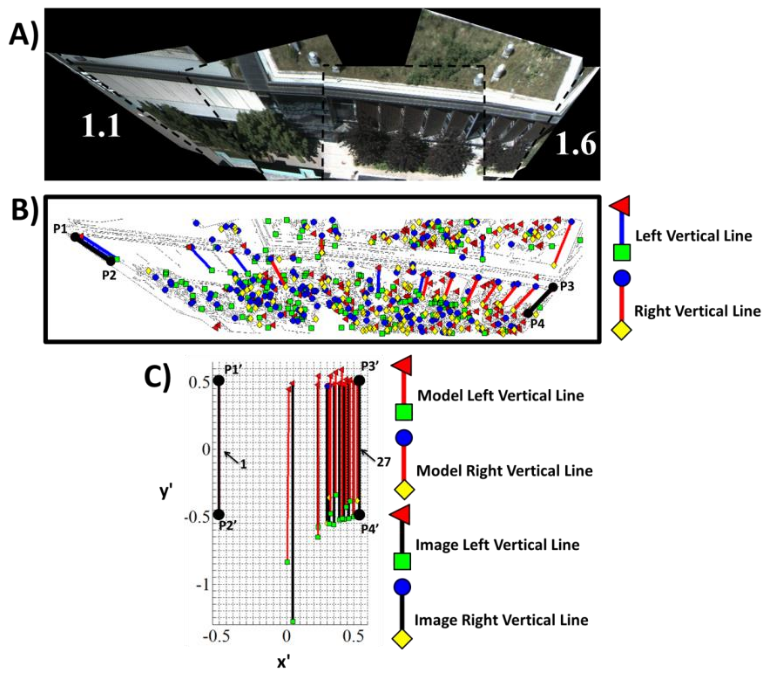


Figure 7.11. Dataset 1: A) 6 images stitched together, B) the Canny edge image and the extracted Vertical Line Features, C) the hash table (M1, (1, 27)), which received the most votes, only the corresponding features are plotted.

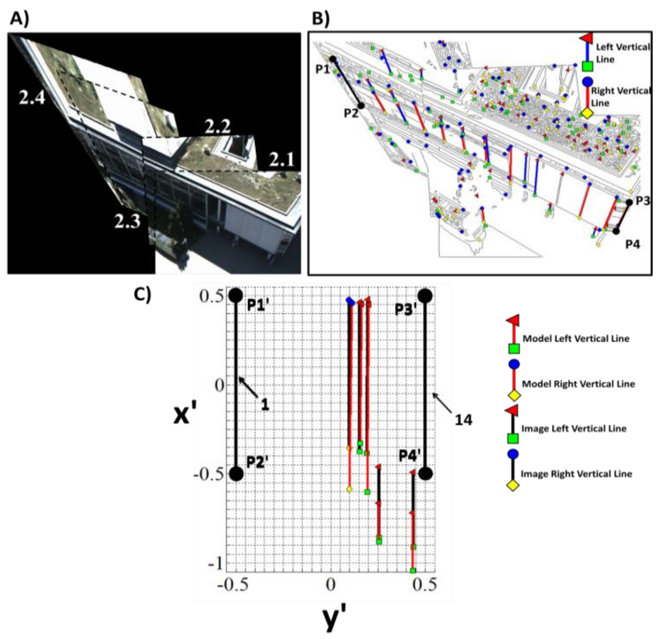


Figure 7.12. Dataset 2: A) Four images stitched together, B) the Canny edge image and the extracted Vertical Line Features, C) the hash table (M4, (1, 14)), which received the most votes, only the corresponding features are plotted.

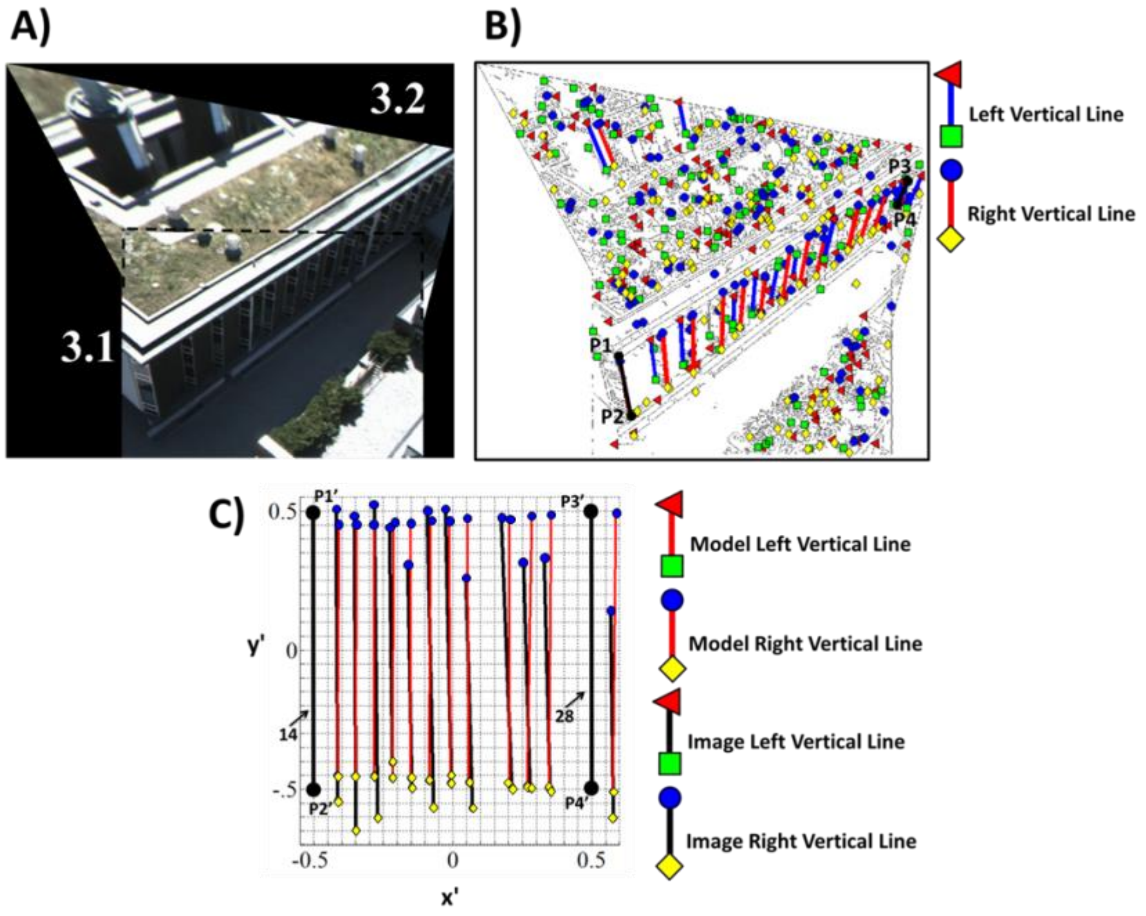


Figure 7.13. Dataset 3: A) 2 images stitched together, B) the Canny edge image and the extracted Vertical Line Features, C) the hash table (M4, (14, 28)), which received the most votes, only the corresponding features are plotted.

7.4.1.1 Verification test

The Vertical Line Features from the synthetic image were matched with the Vertical Line Features from the video frame based on geometric hashing approach. Once the correspondence between the video frame and synthetic image was established, RANSAC (RANdom SAmple Consensus) (Fischler and Bolles, 1981) was used to eliminate mismatches and estimate the geometry robustly, in a similar manner as a Lowe's (1987) verification test using the viewpoint consistency constraint. The parameters of the exterior orientation of the video camera were then determined through a photogrammetric bundle adjustment. Initial parameters of the exterior orientation were determined based on the synthetic images of the 3D model as described earlier.

Figure 7.14 shows the resulting camera positions, 6 images viewing façade 1 (Dataset 1), 4 images viewing façade 2 (Dataset 2), and 2 images viewing façade 3 (Dataset 3).

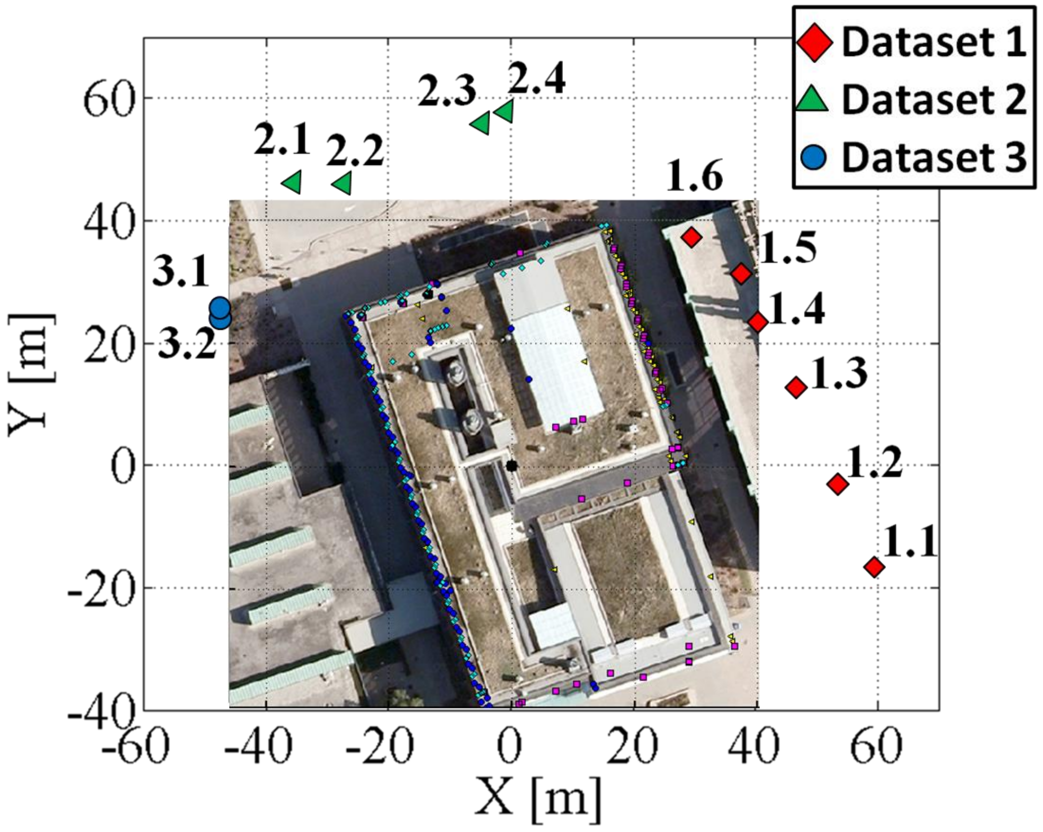


Figure 7.14. Camera positions for Datasets 1, 2, and 3.

Table 7.3 to Table 7.5 show that the bundle adjustment yielded sub-metre positional standard deviations when an image contained at least 4 Vertical Line Features (8 Ground Control Points (GCPs)). This is an improvement from the 2 to 5 metre positional accuracies obtainable by a WAAS-enabled GPS receiver (NSTB, 2006), especially in the vertical component. Further, the photogrammetrically derived orientation was also more accurate than the 2 degree accuracies in roll and pitch, and 5 degree accuracies in heading provided by the UAV's AHRS (Attitude and Heading Reference System).

Table 7.6 to Table 7.8 provide statistics of the object coordinates that were estimated in the respective façade-based bundle adjustments for Datasets 1, 2, and 3, respectively. Dataset 2 (Table 7.7) shows that object coordinates were estimated with sub-metre positional accuracies (root mean square error (RMSE)) when a tie point was visible in 4 images with good image geometry (i.e., wide camera baselines and convergent imagery). Also, standard deviations of the object coordinates were estimated in the bundle adjustment solution. Datasets 1 and 2 shows that when a tie point was visible in only two images, 1 to 3 metre (RMSE) accuracies were obtainable with good image geometry. Dataset 3 shows how drastically the accuracies degrade as the camera baseline shortens.

Table 7.3. Dataset 1: Camera pose standard deviations from the bundle adjustment, with the number of Ground Control Points (GCPs) and number of Tie Points (TP) per image.

Image	**σ_X [m]**	**σ_Y [m]**	**σ_Z [m]**	**σ_ω [°]**	**σ_ϕ [°]**	**σ_κ [°]**	**#GCP**	**#TP**
1.1	7.319	6.965	3.041	8.824	7.403	5.098	2	15
1.2	5.164	7.815	2.345	10.414	6.607	6.137	0	20
1.3	1.685	1.669	0.942	2.741	2.435	2.388	2	11
1.4	0.733	0.806	0.451	1.475	1.252	1.123	13	3
1.5	0.824	0.851	0.502	1.443	1.282	1.062	8	3
1.6	0.745	0.713	0.462	1.259	1.216	1.004	12	3

Table 7.4. Dataset 2: Camera pose standard deviations from the bundle adjustment, with the number of Ground Control Points (GCPs) and number of Tie Points (TP) per image.

Image	**σ_X [m]**	**σ_Y [m]**	**σ_Z [m]**	**σ_ω [°]**	**σ_ϕ [°]**	**σ_κ [°]**	**#GCP**	**#TP**
2.1	0.7015	0.6899	0.4594	0.9774	0.943	0.6252	12	5
2.2	0.7436	0.7718	0.4793	1.1044	0.9816	0.8762	8	12
2.3	2.694	1.2631	1.0184	1.7742	3.5712	0.9513	0	20
2.4	2.3158	1.4204	1.0547	1.8106	2.7787	0.8605	2	18

Table 7.5. Dataset 3: Camera pose standard deviations from the bundle adjustment, with the number of Ground Control Points (GCPs) and number of Tie Points (TP) per image.

Image	**σ_X [m]**	**σ_Y [m]**	**σ_Z [m]**	**σ_ω [°]**	**σ_ϕ [°]**	**σ_κ [°]**	**#GCP**	**#TP**
3.1	0.413	0.289	0.434	0.621	0.637	0.626	16	10
3.2	0.439	0.275	0.520	0.701	0.587	0.680	14	10

Table 7.6. Bundle adjustment statistics for object coordinates - Dataset 1: Number of images a tie point appeared in, number of tie points involved in the calculation, average coordinate standard deviations from the bundle adjustment, average coordinate differences compared to ground truth, and coordinate root mean square error when compared to ground truth.

Overlap	**# TP**	**Average Std. Dev.**			**Average Difference**			**Root Mean Square Error**		
		X [m]	**Y [m]**	**Z [m]**	**X [m]**	**Y [m]**	**Z [m]**	**X[m]**	**Y[m]**	**Z[m]**
3	4	1.930	3.486	4.462	-0.221	-0.051	-0.817	0.461	0.342	1.519
2	18	2.127	1.675	3.294	0.190	0.561	0.513	1.379	2.368	2.264

Table 7.7. Bundle adjustment statistics for object coordinates - Dataset 2: Number of images a tie point appeared in, number of tie points involved in the calculation, average coordinate standard deviations from the bundle adjustment, average coordinate differences compared to ground truth, and coordinate root mean square error when compared to ground truth.

Overlap	**# TP**	**Average Std. Dev.**			**Average Difference**			**Root Mean Square Error**		
		X [m]	**Y [m]**	**Z [m]**	**X [m]**	**Y [m]**	**Z [m]**	**X[m]**	**Y[m]**	**Z[m]**
4	2	0.610	0.429	0.777	-0.106	0.667	0.155	0.266	0.950	0.222
3	7	0.607	0.507	0.872	-0.135	0.662	0.130	0.376	1.505	0.566
2	13	2.369	3.576	4.477	-0.410	0.259	0.284	1.766	1.098	1.316

Table 7.8. Bundle adjustment statistics for object coordinates - Dataset 3: Number of images a tie point appeared in, number of tie points involved in the calculation, average coordinate standard deviations from the bundle adjustment, average coordinate differences compared to ground truth, and coordinate root mean square error when compared to ground truth.

Overlap	**# TP**	**Average Std. Dev.**			**Average Difference**			**Root Mean Square Error**		
		X [m]	Y [m]	Z [m]	X [m]	Y [m]	Z [m]	X[m]	Y[m]	Z[m]
2	10	9.199	5.900	10.582	2.847	-1.966	-3.091	9.747	6.778	11.043

7.4.2 Experiment 2: A map-based absolute localization solution for indoor environments

The following experiments apply the proposed model-based absolute localization solution to estimate the UAV's camera pose with no prior pose information in indoor environments. Images were collected using an RGB camera (640x480 pixel resolution) and processed using the proposed map-based absolute localization algorithm for indoor environments. A panoramic image was captured in 6 locations of the Bergeron Centre: Rooms S1 to S6 as shown in Figure 7.15. The objective of the experiment was to identify the location that each panoramic image was captured by retrieving the panoramic image's corresponding synthetic panoramic image from a database. Corresponding features were also identified between the incoming panoramic image and the synthetic panoramic image retrieved from the database. These corresponding features were used to initialize the camera pose of a model-based tracker in a subsequent process.

The 3D CAD design plan of the Bergeron Centre of Engineering Excellence was used as the known 3D map of the environment. It is the same model that was used in the experiments presented in Sections 5.6 and 6.6.2. This 3D CAD model served two purposes in the proposed approach. Firstly, it provided the necessary level of detail of linear features (vertical lines) for feature matching. Secondly, it provided ground control points to photogrammetrically achieve sub-metre accuracies of the camera's exterior orientation's positional elements.

Table 7.9. Map-based absolute localization threshold values used in the indoor experiment.

Threshold parameter	**Threshold value**
Line length	½ image height
Normalized cross-correlation	0.8

The RGB images were stitched together to generate a single 360 degree panoramic image, this was accomplished in real-time using OpenCV's implementation of (Brown and Lowe, 2007). Figure 7.16 shows the input panoramic images and corresponding synthetic panoramic images. Vertical Line Features extracted from synthetic panoramic images and input panoramic image are shown with green lines and red lines, respectively. Referring to Table 7.9, a value of one half (½) of the panoramic image height was used for the line length threshold in the indoor experiment. This threshold was chosen because it reduced the number of candidate lines to an adequate amount for real-time matching onboard the UAV, while yielding an adequate number of support lines in between the two basis lines to reliably retrieve the correct synthetic image from the database. In the recognition phase, candidate image-model matches were selected if their NCC value was higher than 0.8 and their NCC was the highest in its row and column.

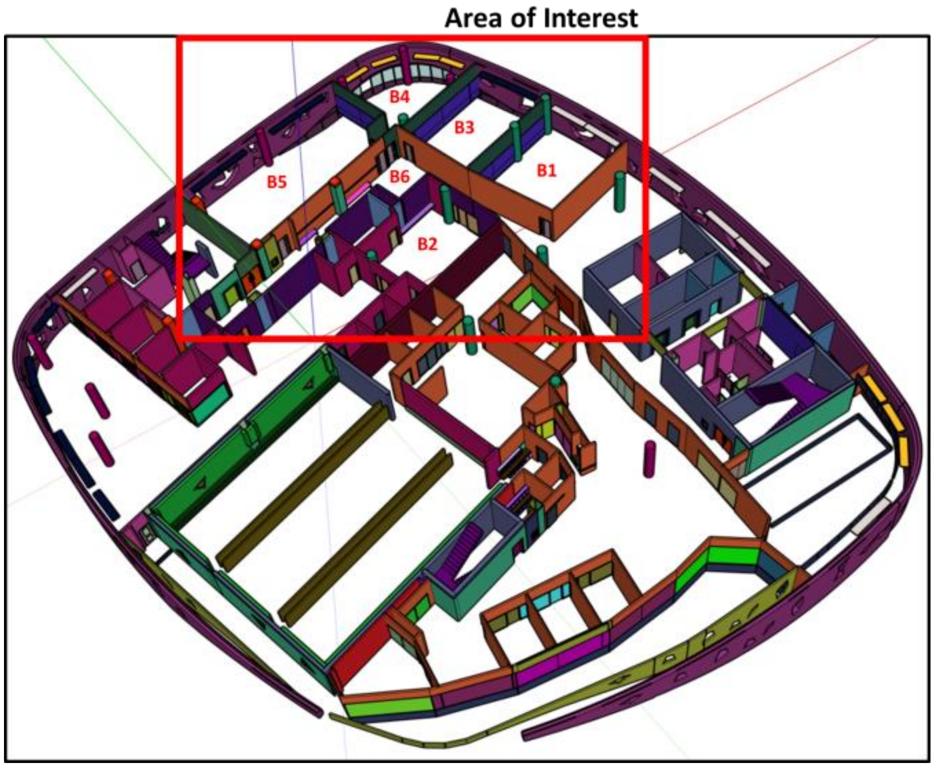


Figure 7.15. Area of interest in the Bergeron Centre for the experiments. Panoramic images were captured in rooms B1, B2, B3, B4, B5, and B6.

Results from using the approach of (Lamdan and Wolfson, 1988), shown in Table 7.10, revealed that using the corner points of Vertical Line Features in geometric hashing's conventional voting strategy produced many false candidates in the recognition phase because the required search area was large. Further correct matches could not be reliably identified. Table 7.11 reveals that matching panoramic and synthetic panoramic images using the proposed approach produced more true positives (i.e., accepted correct matches) than false positives (i.e., accepted incorrect matches), and more true negatives (rejected incorrect matches) than false negatives (i.e., rejected correct matches). The results indicate that there are outlier matches in the list of candidate image-to-model line matches. However, when a panoramic image was considered to match the synthetic panoramic image that produced the largest number of candidate image-to-model line matches, the match was always correct in this dataset.

Table 7.10. Conventional geometric hashing: Truth table of the matching process.

		# of true matches (ground truth)	**# of correctly identified matches**	**# of candidate matches**					
				Synthetic images					
				B1	**B2**	**B3**	**B4**	**B5**	**B6**
Real Images	**B1**	9	1	11	0	1	0	1	0
	B2	17	2	0	105	0	121	7	7
	B3	8	0	2	0	2	0	0	0
	B4	16	6	0	112	0	173	1	2
	B5	16	0	2	3	0	5	35	12
	B6	10	0	2	1	1	0	12	10

Table 7.11. Truth table of the proposed matching process.

		# of true matches (ground truth)	**# of correctly identified matches**	**# of candidate matches**					
				Synthetic images					
				B1	**B2**	**B3**	**B4**	**B5**	**B6**
Real Images	**B1**	9	9	**9**	0	2	0	1	2
	B2	17	12	2	**15**	3	10	7	4
	B3	8	6	0	0	**9**	2	5	1
	B4	16	10	8	3	7	**13**	2	3
	B5	16	13	2	13	5	5	**17**	4
	B6	10	7	2	6	1	3	7	**9**

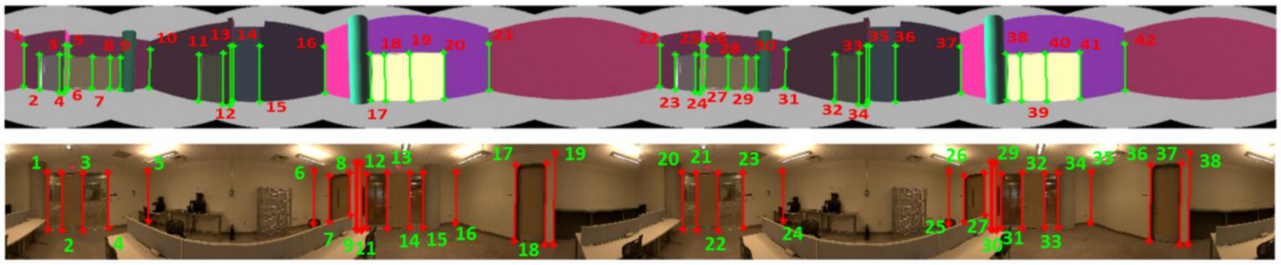
7.5 Concluding remarks

This work utilized geometric hashing to determine correspondence of vertical lines between the UAV's video imagery and the York University 3D campus model without the need of an initial approximate camera pose. The photogrammetric bundle adjustment results showed an improvement in the accuracy of the UAV's navigation solution in dense urban environments. Sub-metre positional accuracies in the object space were achieved when proper viewing geometric configuration, control points and tie points are used. The 3D standard deviation of the estimate position for an image varied from about 10 metres in the worst case to approximately 70 cm in the best case. The 3D standard deviation of the estimated object coordinates ranges from around 15 metres in the worst case to approximately 1 metre in the best case. The average check point difference ranged from 4.6 metres to 0.7 m. The work in this section has contributed to Li-Chee-Ming and Armenakis (2014a).

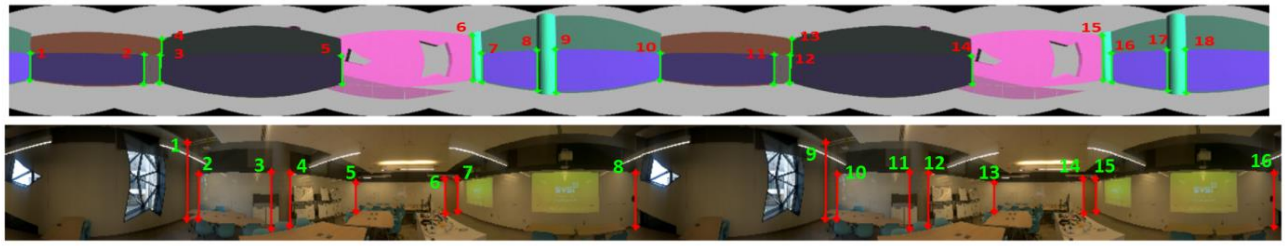
A variant of geometric hashing was proposed that improved performance in matching vertical lines between an incoming panoramic image and a database of virtual panoramic images. A score function was introduced to improve geometric hashing's similarity metric. With enhanced geometric hashing, the correct model image always received the most number of votes. However, in conventional geometric hashing, the correct model was retrieved only 50% of the time. Comparing to ground truth, overall 75% of the correct matches were identified, compared to 11%). Further the number of incorrect matches was reduced by 80% (from 612 incorrectly matched lines to 125). The work in this section has contributed to Li-Chee-Ming and Armenakis (2017b).

Future work may improve upon the limitations of the proposed absolute model-based localization technique. Specifically, the image-to-model matching process is unable to distinguish between identical building facades. In this case, additional sensor data may be used to narrow the search space. For example, the digital compass may indicate the direction the camera is facing. Further, a particular façade will never be retrieved from the database if the façade does not contain vertical lines. This problem may be solved by extracting and searching for other features, for example horizontal lines.

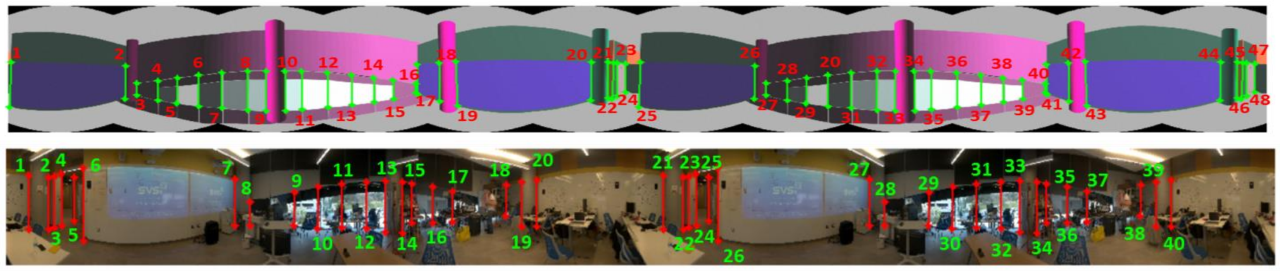
A) B2



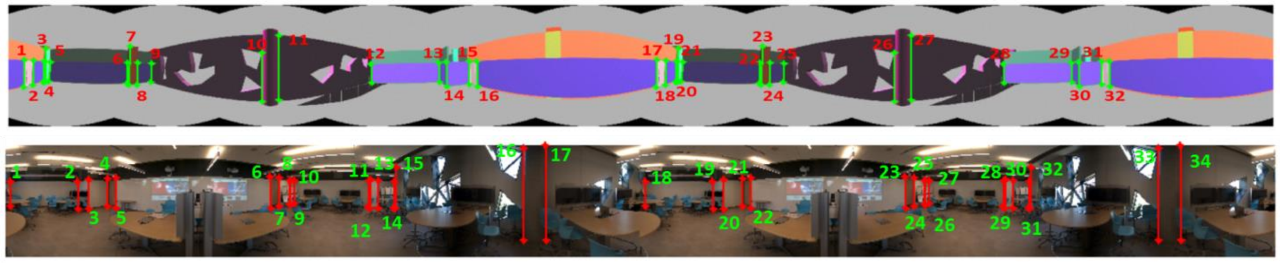
B) B3



C) B4



D) B5



E) B6

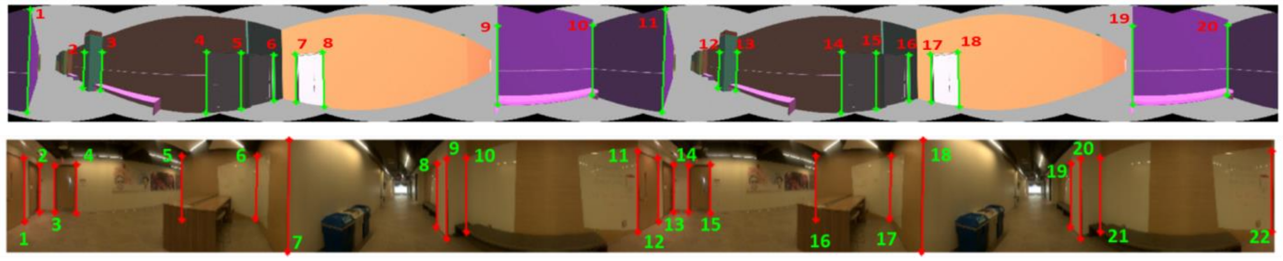


Figure 7.16. Input panoramic images and corresponding synthetic panoramic images captured in the Bergeron Centre in A) Room B2, B) Room B3, C) Room B4, D) Room B5, and E) B6. Vertical Line Features extracted from synthetic panoramic images and input panoramic image are shown with green lines and red lines, respectively.

8 CONCLUSIONS

This dissertation proposed a solution to address the task of fully localizing a monocular camera with respect to a known environment, for the purpose of Unmanned Aerial Vehicle (UAV) navigation in structured environments. This solution first automatically estimated the camera pose with respect to the 3D model of the environment when no prior pose estimate was available. Then, the output of this module initialized a model-based tracker, which is a map-based incremental localization system of the camera using frame-by-frame tracking.

A feasibility study of a state-of-the-art model-based tracker called ViSP revealed its weaknesses in large scales and under challenging imaging conditions. Specifically, ViSP failed when tracking was lost, and needed to be manually re-initialized. This occurred when there was a lack of model features in the camera's field of view, and because of rapid camera motion. Results from outdoor experiments revealed that ViSP achieved similar accuracies to single point positioning solutions obtained from single-frequency (L1) GPS observations – standard deviations ranging from 2-10 metres. These errors are considered to be large, considering the geometric accuracy of the 3D model is 10 to 40 cm. To increase the tracking performance and pose estimation accuracies, ViSP was combined with a simultaneous localization and mapping (SLAM) technique called RGB-D SLAM. The first contribution of this dissertation demonstrated that a partial or a very coarse 3D model could be processed in a SLAM system to simultaneously reconstruct the whole scene or refine the 3D model, with pose estimation benefiting from this reconstruction. By integrating ViSP with RGB-D SLAM, 3 cm mapping accuracies were obtained when using a simple rectangular door model in indoor environments. Additionally, tracking was not lost when the 3D model completely left the camera's field of view. However, experimental results revealed that ViSP was unable to handle occlusions. That is, parts of the 3D model that could not be seen in the image, along with parts of the model located behind the camera, were being projected onto the image. The inability to handle occlusions caused large errors in the camera pose when ViSP tried to match these model lines with lines in the image.

The second contribution of this dissertation proposed a map-based incremental localization algorithm that improves upon the state-of-the-art, specifically aiming to increase pose accuracy and handle occlusions more effectively. This work relied on a classical deterministic non-linear optimization framework, intended to minimize an error between visual features observed in the image and the forward projection of their 3D homologues, using the 3D model. A visual feature was proposed which enabled the representation of environments using sets of straight lines. The novelty of this solution is a more stringent matching process that identifies corresponding linear features from the UAV’s RGB image data and a large, complex, and untextured 3D model. Occlusion handling was performed by the Gazebo virtual environment simulator. Comparative experiments in various environments (i.e., indoor and outdoor) and with various conditions (illumination, specular effects, clutter, etc.) have been carried out, showing the advantages of the different contributions. Specifically, the proposed model-based tracker improved positional accuracies from 10 m to 46 cm in outdoor environments, and improved from an unattainable result to 2 cm positional accuracies in indoor environments.

The issue of the initial localization step was addressed through pose estimation based on a set of initial images formed into a stitched image. The proposed approach lies in the field of template matching methods, where the templates are synthetic images of the environment generated using a priori known 3D CAD model. Again, the Gazebo virtual environment simulator handled occlusions. A first step consisted in storing the views into a hash table to efficiently handle the large search space. Assuming the camera was aligned with the local gravity vector, the method then benefited from vertical line matching to accomplish a registration procedure of the reference views with the set of initial input images. From the most likely view, a pose refinement was finally performed via a best match search in the hash table. Among the contributions, a novel vertical line extraction technique was proposed, which was particularly suited for the context of structured indoor and outdoor environments. A robust lined-based similarity measure was also proposed for the online matching processes of geometric hashing, whose advantage with respect to a classical measure has been demonstrated. The results, performed in various environments (indoor and outdoor) and imaging conditions (synthetic images, real images, cluttered background, specular effects) have shown the efficiency and the robustness of the method to these challenging conditions, as well as its ability to provide a sufficiently precise pose to initialize the tracking step. Results from the outdoor experiment demonstrated that the proposed map-based absolute localization solution produced sub-metre positional accuracies in the object space when proper viewing geometric configuration, control points and tie points are used. Results from the indoor experiment revealed that the proposed enhancement of conventional geometric hashing produced more correct matches - 75% of the correct matches were identified, compared to 11%. Further the number of incorrect matches was reduced by 80%.

Overall this research work has focused on and contributed to the development and implementation of sophisticated methods and data processing algorithms to compensate for the limited performance of the off-the-shelf low-cost, low-accuracy, small light weight UAVs, navigation and remote sensing sensors. Looking ahead, other applications of this unified solution (both initial pose estimation and tracking) can be considered. For example, location-based service applications and applications in augmented reality, or robotic systems involving a monocular camera and a known 3D object.

9 FUTURE WORK

As the structure from motion algorithms require powerful computational capabilities and large data storage facilities, a proposed Software-as-a-Service (SaaS) may be developed on top of the RoboEarth Cloud Engine (RCE) to offload heavy computation and store data to secure computing environments in the Internet cloud, and share and re-use data (www.roboearth.org). The RoboEarth library provides software components commonly used in robotic applications, such as object databases, object recognition and learning models, and a visual SLAM system that is based on a distributed framework. RoboEarth has demonstrated its capabilities with terrestrial robots in indoor environments. For example, an omni-wheel service robot was tasked to serve a drink to a patient in a hospital room. The robot first queried the RoboEarth database for relevant information and downloaded the knowledge previously collected by other robots; such knowledge included object descriptions and instructions on how to complete tasks. The robot then successfully constructed a model of the environment and localized itself, then recognized objects that were downloaded and performed appropriate actions to complete the task (Waibel et al., 2011).

9.1 Cloud-based structure from motion

This section proposes that RoboEarth can also be applied to mapping and tracking applications in outdoor environments by UAVs. Video images captured from the onboard camera are reconstructed using VisualSFM (Wu et al., 2011) and densely matched using SURE (Rothermel et al., 2012). Both are being reformed as an Application-as-a-Service via the RCE. The correct building models are identified from the generated point cloud and the UAV localized itself and mapped the 3D environment in a geodetic coordinate system. The results in Section 6.6.1.1 showed that the estimated position and orientation parameters of the video camera provide increases in accuracy when compared to the UAV autopilot solution, derived from the onboard single frequency GPS receiver and MEMS-IMU.

9.1.1 The RobotEarth Cloud Engine (RCE)

The RoboEarth Cloud Engine (RCE), also called Rapyuta, is an open source robotics Platform-as-a-Service (PaaS) on top of which robotics developers can design robotics Software-as-a-Service (SaaS) applications. Its main difference to similar PaaS frameworks, like the Google App Engine, is that it is specifically designed for multi-process and high-bandwidth robotics applications, such as mapping and object recognition. Further, the RCE is able to launch almost all of the current 3000+ ROS packages in its cloud environment (Mohanarajah et al., 2014). Similar systems include the DAvinCI project (Arumugam et al., 2010), which used ROS to implement a cloud-based parallel implementation of Fast-SLAM (Thrun et al., 2005). Unfortunately, the DAvinCi project is not publicly available. Riazuelo et al. (2013) presented a collaborative mapping system where they moved the computationally intensive bundle adjustment process of the Parallel Tracking and Mapping (PTAM) algorithm (Klein and Murray, 2009) to a server. Heroku (www.heroku.com) provides access to program APIs and sockets, however it does not allow the server to send data to the robot, and does not allow robots to share information through networked computing environments. Rosbridge (Crick et al., 2012) is open source, and enables communication between a robot and one ROS environment in the cloud (Gherardi et al., 2014).

9.1.2 Mapping and tracking in the cloud

This section proposes a UAV navigation system with the objective to track its trajectory and perform 3D mapping of its environment in near-real time. An overview of the system architecture is provided in Figure 9.1. An embedded board with a multicore ARM processor, running a Linux operating system, is used for computations onboard the UAV. The embedded board connects to the RCE through a wireless USB adapter. The RGB Camera ROS node, running on the onboard processor, reads data from the RGB camera and outputs (publishes) an RGB image (topic). Similarly, the Autopilot ROS node reads data from UAV’s sensors (GPS, IMU, magnetometer) and publishes GPS position and AHRS attitude topics. Given the frames produced by RGB Camera and the GPS and attitude data from the Autopilot node, the iSfM ROS node, also running on the onboard processor, estimates the pose of the camera and a 3D point cloud representing the environment. It publishes keyframes to the RCE every time the camera's estimated pose passes a certain threshold relative to the previous keyframe. A keyframe mainly consists of an RGB image, the camera pose, image features and correspondences, and a frame ID. The Map Merger ROS node retrieves maps from the Database and attempts to merge them into a larger map using a modified version of the Iterative Closest Point (ICP) algorithm (Besl and McKay, 1992). Further, the density of the point clouds are increased using SURE (Rothermel et al., 2012), a dense stereo matching software tool based on the semi-global matching (SGM) algorithm (Hirschmüller, 2008). The work in this section has contributed to Li-Chee-Ming and Armenakis (2014b) and Li-Chee-Ming and Armenakis (2014c).

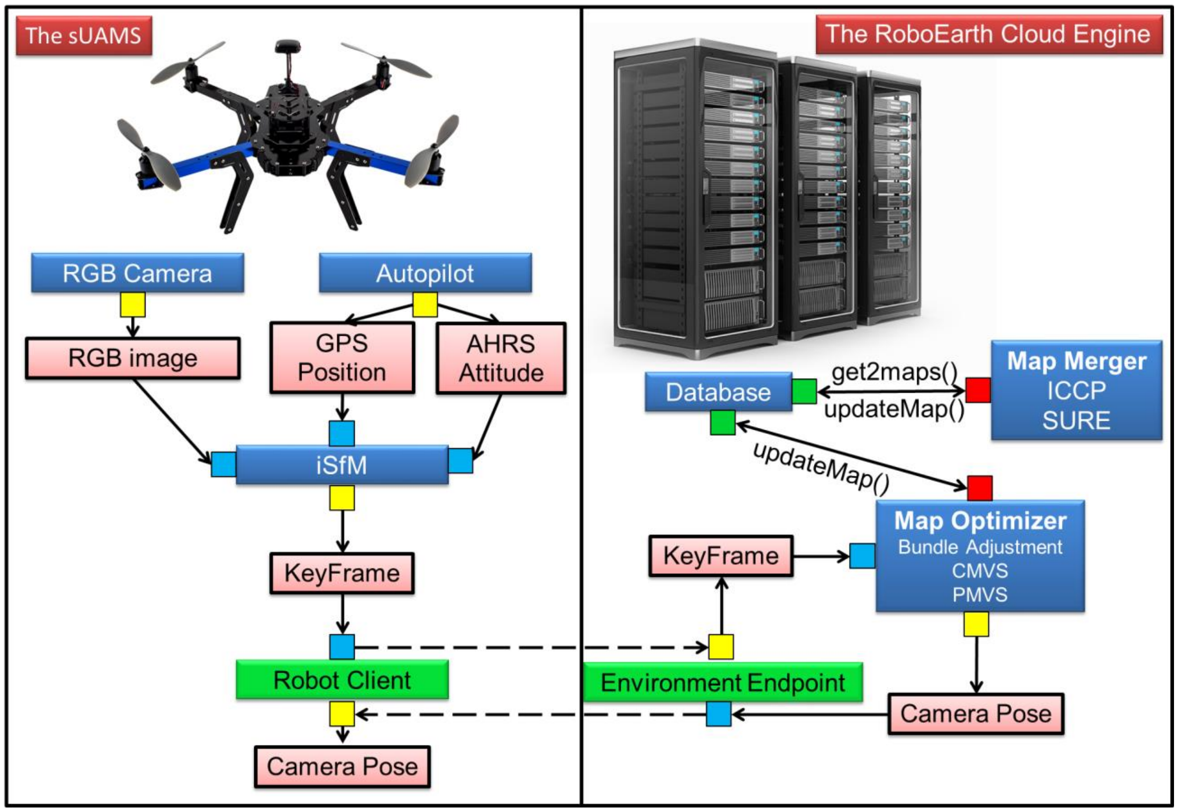


Figure 9.1. Blue rectangles depict ROS nodes, green rectangles for the RCE's endpoints, red rectangles for topics, small yellow squares for topic publishers, small blue squares for topic subscribers, small green squares for service servers, and small red squares for server clients. The direction of the arrows indicates the flow of data from the publishers to the topics and topics to subscribers, and arrows with labels connect service servers to service clients. Notably, this architecture was derived from the developers of the RCE (Gherardi et al., 2014).

REFERENCES

Ahmed M, Khirulmizam S, Rahman R, Raja S, and Salam I. 2009. A review of navigation systems (integration and algorithms). Australian Journal of Basic and Applied Sciences, 3, 943-959.

Applanix. 2017. Direct mapping solutions for UAVs. Retrieved from www.applanix.com/products/dms-uavs.htm. Accessed May 1st, 2017.

Armenakis C, and Sohn G. 2009. ICampus: 3D modeling of York University Campus. Proceedings of the 2009 Conference of the American Society for Photogrammetry and Remote Sensing,Baltimore, Maryland.

Amin D, and Govilkar S. 2015. Comparative study of augmented reality SDKs. Int. J. Computer. Sci, Appl. 5(1):11-26.

Armstrong M, and Zisserman A. 1995. Robust object tracking. In proceedings of Asian Conference on Computer Vision, volume I, pages 58–61.

Arumugam R, Enti V, Baskaran K, and Kumar A. 2010. DAvinCi: A cloud computing framework for service robots. In Proc. IEEE Int. Conf. Robotics and Automation. IEEE, pp. 3084–3089.

Atiya S, and Hager G. 1993. Real-time vision-based robot localization. IEEE Trans. Robotics and Automation, 9(6): 785-800.

Atsuumi K, and Sano M. 2010. Indoor IR azimuth sensor using a linear polarizer. Proceedings of the 2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Campus Science City, ETH Zurich, Switzerland.

Babapour H, Mokhtarzade M, and Valadan Zoej M. 2016. Self-calibration of digital aerial camera using combined orthogonal models. ISPRS Journal of Photogrammetry and Remote Sensing 117: 29–3.

Babapour H, Mokhtarzade M, and Valadan Zoej M. 2017. A novel post-calibration method for digital cameras using image linear features. International Journal of Remote Sensing Vol. 38, Iss. 8-10

Baillard C, Schmid C, Zisserman A, and Fitzgibbon A. 1999. Automatic line matching and 3D reconstruction of buildings from multiple views. Proc. ISPRS Conference on Automatic Extraction of GIS Objects from Digital Imagery, IAPRS, 32: 69-80.

Bar-Shalom Y, and Li X. 1993. Estimation and tracking. Principles, techniques, and software. Artech House, Boston.

Basu S, Essa I, and Pentland A. 1996. Motion regularization for modelbased head tracking. Proc. ICPR, pp. 611-616.

Bay H, Ess A, Tuytelaars T, and Van Gool L. 2008. Speeded-up robust features (SURF). Comput. Vis. Image Underst., vol. 110, pp. 346–359.

Belongie S, Malik J, and Puzicha J. 2002. Shape matching and object recognition using shape contexts. IEEE Trans. PAMI, 24 (4): 509-522.

Benhimane S, Malis E. 2004. Real-time image-based tracking of planes using efficient second-order minimization. IEEE/RSJ Int. Conf. On Intelligent Robots Systems, vol. 943-948, p. 1, Sendai, Japan.

Besl P, and Jain R. 1985. Three-dimensional object recognition. ACM Computing Survey, 17 (1): 75-154.

Besl P, and McKay N. 1992. A method for registration of 3-D shapes. Trans. PAMI. 14(2).

Bisnath S, and Gao Y. 2008. Current state of precise point positioning and future prospects and limitations. In: Proceedings of International Association of Geodesy Symposia: observing our changing earth, vol 133, pp 615–623. Springer, Berlin.

Blake A, and Isard M. 1998. Active contours. Springer Verlag.

Borenstein J, Everett H, and Feng L. 1996. Navigating mobile robots: Systems and techniques. Wellesley, Mass, AK Peters.

Bouthemy P. 1989. A maximum likelihood framework for determining moving edges. IEEE Trans, on Pattern Analysis and Machine Intelligence. 11(5): 499-511.

Brown D. 1966. Decentering distortion of lenses. Photogrammetric Engineering, 32(3), 444-462.

Brown D. 1971. Close-range camera calibration. Photogrammetric Engineering, 4(2):127–140.

Brown M, and Lowe D. 2007. Automatic panoramic image stitching using invariant features. International Journal of Computer Vision, 74(1):59–73.

Brox T, Rosenhahn B, Cremers D, and Seidel H. 2006. High accuracy optical flow serves 3-D pose tracking: exploiting contour and flow based constraints. European Conf. On Computer Vision, ECCV'06, vol. 3952 of LNCS, pp. 98–111, Graz, Austria.

Canny J. 1986. A computational approach to edge detection. IEEE Trans. On Pattern Analysis and Machine intelligence, 8(6):679–698.

Capps C. 2001. Near field or far field. Electronic Design News (EDN), pp. 95–102.

Cham T, Arridhana C, Tan W, Pham M, and Chai L. 2010. Estimating camera pose from a single urban ground-view onmidirectional image and a 2d building outline map. Proc. IEEE Conf. On Computer Vision and Pattern Recognition, San Fransisco, California.

Caruso M. 2000. Applications of magnetic sensors for low cost compass systems. Positioning, Location, and Navigation Symposium (PLANS) 2000 (pp. 177-184). San Diego, CA: Institute of Electrical and Electronics Engineers (IEEE).

Chang X, Yang X, and Zhou T. 2005. MLAMBDA: A modified LAMBDA method for integer least-squares estimation. J. Geodesy, vol.79.

Chatfield A. 1997. Fundamentals of high accuracy inertial navigation. American Institute of Aeronautics and Astronautics, Reston, WA.

Chen Y, and Medioni G. 1992. Object modelling by registration of multiple range images. Image and Vision Computing. 10(3): 145–155.

Chin R, and Dyer C. 1986. Model-based recognition in robot vision. ACM Computing Survey, 18 (1): 67-108.

Choi C, and Christensen H. 2010. Real-time 3D model-based tracking using edge and keypoint features for robotic manipulation. In: International Conference on Robotics and Automation, pp. 4048–4055.

Choi C, and Christensen H. 2012. Robust 3d visual tracking using particle filtering on the special euclidean group: A combined approach of keypoint and edge features. Int. Journal of Robotics Research, 31(4):498– 519.

Clarke T, Wang X, and Fryerj J. 1998. The principal point and CCD cameras. Photogrammetric Record, 16(92), 293-312.

Collins P, Bisnath S, Lahyae F, and Heroux P. 2010. Undifferenced GPS ambiguity resolution using the decoupled clock model and ambiguity datum fixing. NAVIGATION, 57(2), 123-135.

Comport A, Marchand E, Chaumette F. 2003. A real-time tracker for markerless augmented reality. ACM/IEEE Int. Symp. On Mixed and Augmented Reality, ISMAR'03, pp. 36–45, Tokyo, Japan.

Comport A, Marchand E, Pressigout M, Chaumette F. 2006. Real-time markerless tracking for augmented reality: the virtual visual servoing framework. IEEE Trans. On Visualization and Computer Graphics, 12(4):615–628.

Corral-Soto E, Tal R, Wang L, Persad R, Chao L, Chan S, Hou B, Sohn G, and Elder J. 2012. 3D town: The automatic urban awareness project. Proceedings of the 9th Conference on Computer and Robot Vision, Toronto, Ontario, pp. 433–440.

Costa M, and Shapiro L. 2000. 3D object recognition and pose with relational indexing. Computer Vision and Image Understanding, 79(3):364– 407.

Cox I. 1994. Modeling a Dynamic Environment Using a Bayesian Multiple Hypothesis Approach. Artificial Intelligence, 66(2): 311-344.

Crick C, Jay G, Osentoski S, and Jenkins O. 2012. ROS and rosbridge: Roboticists out of the loop. In Proc. Annual ACM/IEEE Int. Conf. Human-Robot Interaction, pp. 493–494.

Dalal N, and Triggs B. 2005. Histograms of oriented gradients for human detection. IEEE Int. Conf. On Computer Vision and Pattern Recognition, pp. 886–893.

David P, and Ho S. 2011. Orientation descriptors for localization in urban environments. Proceedings of the IEEE International Conference on Intelligent Robots and Systems, San Francisco, California.

David P, DeMenthon D, Duraiswami R, and Samet H.2002. SoftPOSIT: Simultaneous pose and correspondence determination. European Conf. On Computer Vision (ECCV), Copenhagen, Denmark, pp. 698-714.

David P, DeMenthon D, Duraiswami R, and Samet H. 2003. Simultaneous pose and correspondence determination using line features. IEEE Int. Conf. On Computer Vision and Pattern Recognition, vol. 2, pp. 424–430, Madison, WI.

Davison A, and Kita N. 2001. Sequential localization and map-building for real-time computer vision and robotics. Robotics and Autonomous Systems, vol. 36, no. 4, pp. 171-183.

Delabarre B, and Marchand E. 2013. Camera localization using mutual information-based multiplane tracking. IEEE/RSJ Int. Conf. On Intelligent Robots and Systems, IROS'2013, Tokyo, Japan.

Dellaert F, Fox D, Burgard W, and Thrun S. 1999. Monte Carlo localization for mobile robots. Proc. IEEE Int'l Conf. Robotics and Automation, pp. 1322-28.

DeMenthon D, and Davis L. 1995. Model-based object pose in 25 Lines of Code. International Journal of Computer Vision, 15(1-2): 123-141.

DeSouza G, and Kak A. 2002. Vision for mobile robot navigation: A survey. IEEE Transaction on Pattern Analysis and Machine Intelligence, 24(2): 237–267.

DJI. 2017. Comparing the DJI Phantom 4 and the Inspire 1. Retrieved from www.drones.specout.com/compare/2-452/DJI-Inspire-1-vs-DJI-Phantom. Accessed May 1, 2017.

Doiphode S, Bakal, J, and Gedam, M. 2016. Survey of indoor positioning measurement, methods and techniques. International Journal of Computer Applications, 140(7): 1-4.

Drummond T, and Cipolla R. 2002. Real-time visual tracking of complex structures. IEEE Trans. On Pattern Analysis and Machine Intelligence, 24(7):932–946.

Duda O, and Hart P. 1972. Use of the Hough transformation to detect lines and curves in pictures. Comm. ACM, pp. 11–15.

Ellum C. 2001. The development of a backpack mobile mapping system. (Master's Thesis). The University of Calgary, Calgary, Alberta.

Ellum C, and El-Sheimy N. 2002. Land-based integrated system for mapping and GIS applications. Survey Review, 36(283).

Endres F, Hess J, Sturm J, Cremers D, and Burgard W. 2014. 3D mapping with an RGB-D camera. IEEE Transactions on Robotics.

Euranto A, Lahdenoja O, and Suominen R. 2014. Model-based tracking initialization in ship building environment. In: University of Turku Technical Reports 2.

Farrell J, and Barth M. 1998. The Global Positioning System & Inertial Navigation. McGraw–Hill.

Federal Aviation Administration. 2017. FAA aerospace forecast: Fiscal Years 2017-2037. Retrieved from: https://www.faa.gov/data_research/aviation/aerospace_forecasts/media/FY2017-37_FAA_Aerospace_Forecast.pdf. Accessed June 18th, 2017.

Ferrari V, Fevrier L, Jurie F, and Schmid C. 2008. Groups of adjacent contour segments for object detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(1), 36–51.

Filonenko V, Cullen C, and Carswell J. 2010. Investigating ultrasonic positioning on mobile phones. Proceedings of the 2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Campus Science City, ETH Zurich, Switzerland.

Fischler N, and Bolles R. 1981. Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. Communication of the ACM, 24(6):381–395.

Foxlin E. 2005. Tracking, auto-calibration, and map-building system. US Patent 6922632.

Frahm J, Fite Georgel P, Gallup D, Johnson T, Raguram R, Wu C, Jen J, Dunn E, Clipp B, Lazebnik S, and Pollefeys M. 2010. Building Rome on a cloudless day. In ECCV, IV: 368–381.

Fraser C. 1997. Digital camera self-calibration. Journal of Photogrammetric Engineering and

Remote Sensing, 52(3), 149-159.

Furukawa Y, and Ponce J. 2010. Accurate, dense, and robust multiview stereopsis. IEEE Trans Pattern Analysis and Machine Intelligence (PAMI). 32(8): 1362–1376.

Furukawa Y, Curless B, Seitz S, Szeliski R. 2010. Towards Internet-scale multi-view stereo. In CVPR10.

Gavrilla D, and Groen D. 1992. 3D object recognition from 2D images using geometric hashing. Pattern Recognition. Lett, 13(4): 263-278.

Gavrilla D, and Philomin V. 1999. Real-time object detection for smart vehicles Proc. 13th Int. Conf. Of Computer Vision, Fort Collins, Colorado, USA, pp. 87-93.

Gennery D. 1992. Visual tracking of known three-dimensional objects. Int. J. Of Computer Vision, 7(3):243–270.

Geva A, Briskin G, Rivlin E, Rotstein H. 2015. Estimating camera pose using Bundle Adjustment and Digital Terrain Model constraints. IEEE International Conference on Robotics and Automation (ICRA), pp. 4000–4005.

Gherardi L, Hunziker D, and Mohanarajah G. 2014. A software product line approach for configuring cloud robotics application. In Proc. IEEE Cloud 2014.

Gini R, Pagliari D, Passoni D, Pinto L, Sona G, and Dosso P. 2013. UAV photogrammetry: block triangulation comparisons. ISPRS Archives, 157–162.

Godin, G., M. Rioux and R. Baribeau. 1994. Three-dimensional registration using range and intensity information. Proc. SPIE Videometrics III, vol. 2350: 279–290.

Gold S, Rangarajan A, Lu C, Pappu S, and Mjolsness E. 1998. New algorithms for 2D and 3D point matching: Pose estimation and correspondence. Pattern Recognition, 31(8): 1019-1031.
Goldstein H. 1980. Classical mechanics. Cambridge, MA: Addison-Wesley.

Green W, Oh P, Sevcik K, and Barrows G. 2003. Autonomous landing for indoor flying robots using optic flow. In Proc. Of IMECE International Mechanical Engineering Congress, volume 2, pp.1341–1346.

Green W, Oh P, and Barrows G. 2004. Flying insects inspired vision for autonomous aerial robot maneuvers in mear-earth environments. In Proc. Of IEEE Int’l Conf. On Robotics and Automation (ICRA), pages 2347–2352.

Grimson W, and Huttenlocher D. 1988. On the sensitivity of the Hough Transform for object recognition, MIT AI Memo No. 1044.

Gu C, and Ren X. 2010. Discriminative mixture-of-templates for viewpoint classification. European Conf. On Computer Vision, pp. 408–421. Heraklion, Crete, Springer.

Habib A, and Alruzouq R. 2004. Line-based modified iterated Hough Transform for automatic registration of multi-source imagery. The Photogrammetric Record 19 (105): 5–21.

Habib, A, Asmamaw A, Kelley D, and May M. 2000. Linear features in photogrammetry. Geodtic Science and Surveying, 9: 3–24.

Habib A, and Kelley D. 2001. Automatic relative orientation of large scale imagery over urban areas using modified iterated Hough Transform. ISPRS Journal of Photogrammetry and Remote Sensing 56 (1): 29–41.

Habib A, Lin H, and Morgan M. 2003. Autonomous space resection using point- and line-based representation of FREE-FORM control linear features. The Photogrammetric Record, 18 (103): 244–258.

Habib A, Morgan Kim M, and Cheng R. 2004. Linear features in photogrammetric activities. ISPRS Congress, Istanbul, Turkey, 610 p.

Habib, A., M. Morgan, and Y.-R. Lee. 2002. Bundle adjustment with self–calibration using straight lines. The Photogrammetric Record 17 (100): 635–650.

Haines E. 1994. Point in Polygon Strategies. Graphics Gems, Ed. Heckbert, Paul. Academic Press.

Harris C, and Stephens M. 1988. A combined corner and edge detector. In Alvey Vision Conference, 147–152.

Helle S, Kaustinen M, Korhonen S, and Lehtonen T. 2013. Indoor localization solutions for a marine industry augmented reality tool. University of Turku, Technical Reports, No 1.

Harris C. 1992. Tracking with rigid objects. MIT Press.

Harris C, and Stephens M. 1988. A combined corner and edge detector. Alvey Conference, pp. 147–151, Manchester.

Hasan A, Samsudin K, Ramli A, Azmir R, and Ismaeel S. 2009. A review of navigation systems (integration and algorithms). Australian Journal of Basic and Applied Sciences, 3(2), 943-959.

Hinterstoisser S, Lepetit V, Ilic S, Fua P, and Navab N. 2010. Dominant orientation templates for real-time detection of texture-less objects. IEEE Int. Conf. On Computer Vision and Pattern Recognition, pp. 2257–2264, San Francisco.

Hirschmüller, H. 2008. Stero processing by semi-global matching and mutual information. IEEE Transactions on Pattern Analysis and Machine Intelligence 30, pp. 328-341.

Hofmann-Wellenhof B, Lichtenegger H, and Collins J. 2001. Global Positioning System: Theory and Practice. 5th ed. New York: Springer Verlag Wien.

Holzer S, Hinterstoisser S, Ilic S, and Navab N. 2009. Distance transform templates for object detection and pose estimation. IEEE Int. Conf. On Computer Vision and Pattern Recognition, pp. 1177–1184, Miami, FL.

Horn, B, and Schunck, B. 1981. Determining optical flow. Artif. Intell. 17(1–3), 185–203.

Hrabar S, Sukhatme G, Corke P, Usher K, and Roberts J. 2005. Combined optic-flow and stereo-based navigation of urban canyons for a UAV. In Proc. Of IEEE Int'l Conf. Of Intelligent Robots and Systems (IROS), pages 3309–3316.

Huttenlocher D, Klanderman G, and Rucklidge W. 1993. Comparing images using the Hausdorff Distance. IEEE Trans. Pattern Analysis and Machine Intelligence, 15(9): 805-863.

IGI. 2017. TERRAcontrol - GNSS/IMU navigation system. Retrieved from www.igi-systems.com/terracontrol.html. Accessed May 1st, 2017

IGS. 2017. International GNSS Service. Retrieved from www.igscb.jpl.nasa.gov. Accessed May 1st, 2017.

Isard M, and Blake A. 1998. Condensation – conditional density propagation for visual tracking. Int. J. Computer Vision, 29(1):5–28.

Jazwinski A. 1970. Stochastic process and filtering theory. New York: Academic Press.

Jekeli C. 2001. Inertial navigation systems with geodetic applications. Walter de Gruyter, New York, NY, 352 p.

Jones S, Andresen C, and Crowley J. 1997. Appearance based processes for visual navigation. In Proc. Of IEEE Int'l Conf. On Intelligent Robots and Systems (IROS), pages 551–557.

Jung J**,** Sohn G, Bang K**,** Wichmann A**,** Armenakis C, and Kada M. 2016. Matching aerial images to 3D building models using context-based geometric hashing. S*ensors*, 16(6):93.

Jurie F, and Dhome M. 2001. A simple and efficient template matching algorithm. Proc. ICCV, pp. 544-549.

Kabuka M, and Arenas A. 1987. Position verification of a mobile robot using standard pattern. IEEE J. Robotics and Automation, 3(6): 505-516.

Kalman R. 1960. A new approach to linear filtering and prediction problems. Transactions of the ASME, Ser. D, Journal of Basic Engineering, 82, 34-45.

Khoshelham, K. 2010. Automated localization of a laser scanner in indoor environments using planar objects. Proceedings of the 2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Campus Science City, ETH Zurich, Switzerland.

Klein G, and Murray D. 2006. Full-3d edge tracking with a particle filter. British Machine Vision Conf., vol. 3, pp. 1119–1128, Edinburgh.

Koller D, Daniilidis K, and Nagel H. 1993. Model-based object tracking in monocular image sequences of road traffic scenes. Int. Journal of Computer Vision, 10(2):257–281.

Kong X. 2000. Inertial navigation system algorithms for low cost IMU (Ph.D. Dissertation). The University of Sydney, Sydney, New South Wales.

Kosaka A, and Kak A. 1992. Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties. Computer Vision, Graphics, and Image Processing—Image Understanding, 56(3): 271-329.

Krotkov E. 1989. Mobile robot localization using a single image. Proc. IEEE Int’l Conf. Robotics and Automation, pp. 978-983.

Kubik K. 1991. Relative and absolute orientation based on linear features. ISPRS Journal of Photogrammetry and Remote Sensing, Vol. 46, pp. 199-204.

Kuipers J. 1999. Quaternions and rotation sequences. Princeton, New Jersey: Princeton University Press.

Kümmerle R, Grisetti G, Strasdat H, Konolige K, and Burgard W. 2011. G2o: A general framework for graph optimization. In Proc. Of the IEEE Intl. Conf. On Robotics and Automation (ICRA), Shanghai, China.

Kyrki V, and Kragic D. 2011. Tracking rigid objects using integration of model-based and model-free cues. In: Machine Vision and Applications 22.2, pp. 323–335.

Lague D, Brodu N, and Leroux J. 2013. Accurate 3D comparison of complex topography with terrestrial laser scanner: application to the Rangitikei canyon (N-Z). ISPRS Journal of Photogrammetry and Remote Sensing. 82: 10–26.

Lahdenoja O, Suominen R, Säntti T, and Lehtonen T. 2015. Recent advances in monocular model-based tracking: A systematic literature review. University of Turku Technical Reports, No. 8.

Lamdan Y, and Wolfson H. 1988. Geometric hashing: A general and efficient model-based recognition scheme. Proc. Second Int'l Conf. Computer Vision, pp. 238-249.

Lamdan Y, and Wolfson H. 1989. On the error analysis of geometric hashing. NYU Robotics Research Report 213.

Lamdan Y, Schwartz J, and Wolfson H. 1990. Affine invariant model-based object recognition. IEEE Transactions on Robotics and Automation, 6(1):578–589.

Latecki L, Lakamper R, and Eckhardt U. 2000. Shape descriptors for non-rigid shapes with a single closed contour. Proc. IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head Island, SC., pp. 424-429.

Lee S, and Song J. 2007. Mobile robot localization using infrared light reflecting landmarks. Proceedings of the International Conference on Control, Automation and Systems (ICCAS'07), pp. 674–677.

Legat K. 2006. Approximate direct georeferencing in national coordinates. Journal of Photogrammetry and Remote Sensing, 60(4), 239-255.

Lepetit V, and Fua P. 2005. Monocular model-based 3d tracking of rigid objects: A survey. In: Foundations and trends in computer graphics and vision, pp. 1–89.

Lerma J, and Cabrelles M. 2007. A review and analyses of plumb-line calibration. The Photogrammetric Record, 22(118): 135-150.

Leung K, Clark C, and Huisson J. 2008. Localization in urban environments by matching ground level video images with an aerial image. Proc. IEEE Int. Conf. On Robotics and Automation, pp. 551-556.

Lewis J. 1995. Fast normalized cross-correlation, Vision Interface.

Li-Chee-Ming, J, Armenakis C. 2013. Determination of UAS trajectory in a known environment from FPV video. UAV-g (Unmanned Aerial Vehicles in Geomatics) Conference, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, ISPRS ICWG I/V, Rockstock, Germany.

Li-Chee-Ming J, Armenakis C. 2014a. Feasibility study for pose estimation of small UAS in known 3D environment using geometric hashing. Photogrammetric Engineering & Remote Sensing. Vol. 80, No. 12, pp. 1117–1128.

Li-Chee-Ming J, Armenakis C. 2014b. Generation of dense 3D point clouds using a small quadcopter. GEOMATICA, 68(4):319-330.

Li-Chee-Ming J, Armenakis C. 2014c. Feasibility study of using ROBOEARTH's cloud computing for rapid mapping and tracking with small unmanned aerial systems. ISPRS TC I Symposium 2014, United States, Colorado, November, Inter. Archives of ISPRS, Vol XL-1, pp 219-226.

Li-Chee-Ming J, Armenakis C. 2015. A feasibility study on using ViSP'S 3D model-based tracker for UAV pose estimation in outdoor environments. UAV-g 2015, ISPRS Archives, Vol XL-1/W4, pp 329-335.

Li-Chee-Ming J, Armenakis C. 2016. Augmenting ViSP's 3D model-based tracker with RGB-D SLAM for pose estimation in indoor environments. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., ISPRS Congress, Prague, Czech Republic, Vol XLI-B1, 925-932.

Li-Chee-Ming J, Armenakis C. 2017a. UAV navigation system using line-based sensor pose estimation. Geo-spatial Information Science Special Issue on Visual Information Processing for Unmanned Aerial Vehicles (UAVs). (Under review)

Li-Chee-Ming J, Armenakis C. 2017b. Matching real and synthetic panoramic images using a variant of geometric hashing. ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., IV-1-W1, 199-206.

Liang B, and Pears N. 2002. Visual navigation using planar homographies. In Proc. Of IEEE Int'l Conf. On Robotics and Automations (ICRA), volume 1, pages 205–210.

Liebelt J, Schmid C, and Schertler K. 2008. Viewpoint-independent object class detection using 3D feature maps. IEEE Int. Conf. On Computer Vision and Pattern Recognition, Anchorage, Alaska.

Liu Y, and Huang T.S. 1991. Determining straight line correspondences from intensity images. Pattern Recognition, 24(6): 489–504.

Long T, Jiao W, and Wang W. 2011. Geometric rectification using feature points supplied by straight-Lines. Procedia Environmental Sciences 11: 200–207.

Lowe D. 1987. Three-dimensional object recognition from single two-dimensional images. Artificial Intelligence, 31(3):355–394.

Lowe. D. 1992. Robust model-based motion tracking trough the integration of search and estimation. Int. Journal of Computer Vision, 8(2):113–122.

Lowe D. 1999. Object recognition from local scale-invariant features. 7th Conference on Computer Vision, pp.1150-1157.

Lowe D. 2004. Distinctive image features from scale-invariant keypoints. Int. Journal of Computer Vision, 60(2):91–110.

Lucas B, and Kanade T. 1981. An iterative image registration technique with an application to stereo vision. In: Proc. Of DARPA Image Understanding Workshop, pp. 121–130.

Marchand E, Bouthemy P, Chaumette F, and Moreau V. 1999. Robust real-time visual tracking using a 2D-3D model-based approach. IEEE Int. Conf. On Computer Vision, ICCV'99, vol. 1, pp. 262–268, Kerkira, Greece.

Marchand E, and Chaumette F. 2002. Virtual visual servoing: a framework for real-time augmented reality. EUROGRAPHICS'02 Conf. Proceeding, vol. 21(3) of Computer Graphics Forum, pp. 289–298, Saarebrücken, Germany.

Matas J, and Galambos C, and Kittler J. 2000. Robust detection of lines using the progressive probabilistic Hough Transform. CVIU 78 1, pp 119-137.

Matsumoto Y, Inaba M, and Inoue H. 1996. Visual navigation using view sequenced route representation. In Proc. Of IEEE Int'l Conf. On Robotics and Automation (ICRA), volume 1, pages 83–88.

Matthies L, Maimone M, Johnson A, Cheng Y , Willson R, Villalpando C, Goldberg S, Huertas A, Stein A, and Angelova A. 2007. Computer vision on Mars. International Journal of Computer Vision, 75(1): 67–92.

Mautz R. 2012. Indoor positioning technologies. Habilitation dissertation, ETH Zurich.

Mautz R, and Ochieng W. 2007. A robust indoor positioning and auto-localization algorithm. Journal of Global Positioning Systems, 6(1): 38–46.

Maye O, Schaeffner J, and Maaser M. 2006. An optical indoor positioning system for the mass market. Proceedings of the 3rd Workshop on Positioning, Navigation and Communication (WPNC'06), Hanover, Germany, IEEE Xplore, pp. 111–115.

Meierhold N, Bienert A, and Schmich A. 2008. Line-based referencing between images and laser scanner data for image based point cloud interpretation in a cad-environment. In: ISPRS Archives, Vol. XXXVII-B5, pp.437-443.

Mejias L, Saripalli S, Sukhatme G, and Cervera P. 2005. Detection and tracking of external features in an urban environment using an autonomous helicopter. In Proc. Of IEEE Int'l Conf. On Robotics and Automation (ICRA), pages 3972–3977.

Microsoft. 2016. Microsoft indoor localization competition. Retrieved from: www.microsoft.com/en-us/research/event/microsoft-indoor-localization-competition-ipsn-2016/. Accessed May 1st, 2017.

Mohanarajah G, Hunziker D, Waibel M, and D'Andrea R. 2014. Rapyuta: A cloud robotics platform. IEEE Transactions on Automation Science and Engineering.

Morel J, and Yu G. 2009. ASIFT: a new framework for fully affine invariant image comparison. SIAM J. Imaging Sci. 2: 438–469.

Mulawa D, and Mikhail E. 1988. Photogrammetric treatment of linear features. International Archives of Photogrammetry and Remote Sensing 27, (B3): 383–393.

Mulloni A, Wgner D, Schmalstieg D, and Barakonyi I. 2009. Indoor positioning and navigation with camera phones. Pervasive Computing, IEEE, vol. 8, pp. 22–31.

Nakamura T, and Asada M. 1995. Motion sketch: Acquisition of visual motion guided behaviors. Proc. 14th Int'l Joint Conf. Artificial Intelligence, vol. 1, pp. 126-132.

National Satellite Test Bed (NSTB). 2006. WAAS PAN Report. Retrieved from: www.nstb.tc.faa.gov/REPORTS/waaspan17.pdf. Assessed May 1st, 2017.

Nocedal J, Wright S. 2000. Numerical optimization. Springer.

NIMA. 2000. NIMA technical report TR8350.2: Department of defense world geodetic system 1984, its definition and relationships with local geodetic systems. Third Edition.

NRCAN (Natural Resources Canada). 2017. Magnetic declination calculator. Retrieved from www.geomag.nrcan.gc.ca/calc/mdcal-en.php. Accessed May 1st, 2017.

Ohno T, Ohya A, and Yuta S. 1996. Autonomous navigation for mobile robots referring rre-recorded image sequence. In Proc. Of IEEE Int'l Conf. On Intelligent Robots and Systems (IROS), volume 2, pages 672–679.

Ollero A, Ferruz J, Caballero F, Hurtado S, and Merino L. 2004. Motion compensation and object detection for autonomous helicopter visual navigation in the COMETS system. In Proc. Of IEEE Int'l Conf. On Robotics and Automation (ICRA), pages 19–24.

Olson C, Huttenlocher D. 1997. Automatic target recognition by matching oriented edge pixels. IEEE Trans. On Image Processing, 6(1):103–113.

Open Source Robotics Foundation. 2017. Gazebo, robot simulation made easy. Retrieved from www.gazebosim.org/. Accessed May 1st, 2017.

Ozuysal M, Calonder M, Lepetit V, Fua P. 2010. Fast keypoint recognition using random ferns. IEEE Trans. On PAMI, 32(3):448–461.

Ozyagcilar T. 2011. Implementing a tilt-compensated eCompass using accelerometer and magnetometer sensors. Freescale Semiconductor Application Note. Retrieved from www.freescale.com/files/sensors/doc/app_note/AN4248.pdf

Payet N, and Todorovic S. 2011. From contours to 3d object detection and pose estimation. IEEE Int. Conf. On Computer Vision, pp. 983–990, Barcelona, Spain.

Pears N and Liang B. 2001. Ground plane segmentation for mobile robot visual navigation. In Proc. Of IEEE Int'l Conf. On Intelligent Robots and Systems (IROS), pages 1513–1518, 2001.

Petit A. 2014. Robust visual detection and tracking of complex objects: applications to space autonomous rendezvous and proximity operations. Ph.D. Dissertation, Computer Vision and Pattern Recognition, Université Rennes.

Pressigout M, and Marchand E. 2004. Model-free augmented reality by virtual visual servoing. IAPR Int. Conf. On Pattern Recognition, ICPR'04, vol. 2, pp. 887–891, Cambridge, UK.

Prisacariu V, Kähler O, Murray D, and Reid I. 2013. Simultaneous 3D tracking and reconstruction on a mobile phone. IEEE Int. Symp. On Mixed and Augmented Reality, ISMAR'13.

Prisacariu V, and Reid I. 2012. Pwp3d: Real-time segmentation and tracking of 3D objects. Int. Journal of Computer Vision, 98(3):335–354.

Pulli K. 1999. Multiview registration for large data sets. Proc. 2nd Int'l Conf. 3-D Digital Imaging and Modeling, Ottawa, 160–168.

Ressl C. 2002. The impact of conformal map projections on direct georeferencing. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 34(3A), 283–288.

Reitmayr G, and Drummond T. 2006. Going out: Robust model-based tracking for outdoor augmented reality. In: International Symposium on Mixed and Augmented Reality (ISMAR), pp. 109–118.

Riazuelo L, Civera J, and Montiel J. 2013. C2tam: A cloud framework for cooperative tracking and mapping. Robotics and Autonomous Systems.

Rigoutsos I, and Hummel R. 1995. A Bayesian approach to model matching with geometric hashing. Computer Vision and Image Understanding, 61(7):11–26. Rothwell, C.A., A.

Roberts K.S. 1988. A new representation for lines. IEEE Proceedings of Computer Vision and Pattern Recognition, pp. 635-640.

Rodrigues J, Kim J, Furukawa M, Xavier J, Aguiar P, and Kanade T. 2012. 6D pose estimation of textureless shiny objects using random ferns for bin-picking. Pp. 3334–3341, Algarve, Portugal.

Rothermel M, Wenzel K, Fritsch D, and Haala N. 2012. SURE: photogrammetric surface reconstruction from imagery. Proceedings LC3D Workshop, Berlin.

Rothwell C, Zisserman A, Forsyth D, and Mundy J. 1995. Planar object recognition using projective shape representation. Int. Journal of Computer Vision, 16:57-99.

Saeedi P, Lawrence P, and Lowe D. 2006. Vision-based 3D trajectory tracking for unknown environments. IEEE Transactions on Robotics, 22(1):119–136.

Schenk, T. 2004. From point-based to feature-based aerial triangulation. ISPRS Journal of Photogrammetry and Remote Sensing, 58 (5–6): 315–329.

Schwarz K, and Wei M. 2000. INS/GPS integration for geodetic applications (Lecture Notes). Department of Geomatics Engineering, University of Calgary, Calgary, Alberta.

Schweinzer H, and Syafrudin S. 2010. LOSNUS: An ultrasonic system enabling high accuracy and secure TDoA locating of numerous devices. Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN), Campus Science City, ETH Zurich, Switzerland.

Schwermann R. 1995. Geradengestützte bildorientierung in der nahbereichsphotogrammetrie. Dissertation, Veröffentlichung des Geodätischen Instituts der Rheinisch-Westfälischen Technischen Hochschule Aachen, Nr 52.

Se S, Lowe D, and Little J. 2002. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. The International Journal of Robotics Research, 21(8):735–758.

Se S, Lowe D, and Little J. 2005. Vision-based global localization and mapping for mobile robots. IEEE Transactions on Robotics, 21(3):364–375.

Seepersad G, and Bisnath S. 2013. Integrity monitoring in precise point positioning. Proc. ION GNSS 2013, Institute of Navigation, Nashville, Tennessee, USA.

Seo B, Park J, and Park J. 2011. 3D visual tracking for mobile augmented reality applications. In: International Conference on Multimedia and Expo (ICME), pp. 1–4.

Shaker A. 2004. The line based transformation model (LBTM): A new approach to the rectification of high resolution satellite imagery. The International Archives of Photogrammetry, Remote Sensing, and Spatial Information Sciences, Istanbul, XXXV (3).

Shi J, and Tomasi C. 1994. Good features to track. IEEE Int. Conf. On Computer Vision and Pattern Recognition, CVPR’94, pp. 593–600, Seattle, Washington.

Shotton J, Blake A, and Cipolla R. 2005. Contour-based learning for object detection. IEEE Int. Conf. On Computer Vision, pp. 503–510.

Simmons R, and Koenig S. 1995. Probabilistic robot navigation in partially observable environments. Proc. Int’l Joint Conf. Artificial Intelligence, pp. 1080-1087.

Skaloud J. 1999. Optimizing georeferencing of airborne survey systems by INS/DGPS (Ph.D. Dissertation). The University of Calgary, Calgary, Alberta.

Skaloud J, and Legat K. 2007. Theory and reality of direct georeferencing in national coordinates. Journal of Photogrammetry and Remote Sensing, 63(2), 272-282.

Skylogic Research. 2016. 2016 Drones in the channel and drone buyer research results. Retrieved from: www.slideshare.net/ColinSnow/2016-drone-buyer-research-results. Accessed June 19th, 2017.

Skrypnyk I, and Lowe D. 2004. Scene modelling, recognition and tracking with invariant image features. ISMAR,pp. 110-119.

Smith P, Reid I, and Lucas A. 2006. Real-time monocular SLAM with straight lines. Proceedings of the British Machine Vision Conference (BMVC 2006), Edinburgh.

Treiber, M. Introduction to object recognition. Springer-Verlag London, Limited, 220 p.

Stark M, Goesele M, and Schiele B. 2010. Back to the future: Learning shape models from 3D cad data. British Machine Vision Conf., Aberystwyth, Wales.

Strecha C, Bronstein A, Bronstein M, and Fua P. 2012. LDAHash: improved matching with smaller descriptors. IEEE Trans. Pattern Anal. Mach. Intell. 34: 66–78.

Sugihara K. 1988. Some location problems for robot navigation using a single camera. Computer Vision, Graphics, and Image Processing, vol. 42, pp. 112-129.

Sundareswaran V, and Behringer R. 1998. Visual servoing-based augmented reality. In IEEE Workshop on Augmented Reality, San Fransisco.

Szarmes M, Lachapelle G, and Cannon M. 1997. Development of a low cost real-time twin-antenna GPS heading system. Contract Report for Defence Research Establishment Ottawa. Department of National Defence. Canada.

Takasu T, and Yasuda A. 2009. Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB. International Symposium on GPS/GNSS, International Convention Center Jeju, Korea.

Tamaazousti M, Gay-Bellile V, Collette S, Bourgeois S, and Dhome M. 2011. Nonlinear refinement of structure from motion reconstruction by taking advantage of a partial knowledge of the environment. Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pp. 3073–3080.

Tamadazte B, Marchand E, Dembélé S, and Le Fort-Piat N. 2010. CAD model-based tracking and 3D visual-based control for MEMS micro-assembly. The International Journal of Robotics Research, 29(11):1416–1434.

Teulière C, Marchand E, and Eck L. 2010. Using multiple hypothesis in model-based tracking. IEEE Int. Conf. On Robotics and Automation, ICRA'10, pp. 4559–4565, Anchorage, Alaska.

Teunissen, P. 1995. The least-square ambiguity decorrelation adjustment: A method for fast GPS ambiguity estimation. Journal of Geodesy, 70, 65-82.

Thrun S. 2000. Probabilistic algorithms in robotics. Technical report CMU-CS-00-126, Carnegie Mellon University.

Thrun S, Burgard W, and Fox D. 2005. Probabilistic robotics (intelligent robotics and autonomous agents). The MIT Press.

Titterton D, and Weston J. 1997. Strapdown inertial navigation technology. IEE Radar, Sonar, Navigation and Avionics Series, 5. Stevenage, United Kingdom: Peter Peregrinus Ltd.

Tommaselli A, and Lugnani J. 1998. An alternative mathematical model to collinearity equations using straight features. International Archives of Photogrammetry and Remote Sensing, 27 (B3): 765–774.

Torres-Solis J, Falk T, and Chau T. 2010. A review of indoor localization technologies: Towards navigational assistance for topographical disorientation. Ambient Intell. 2010, 3, 51–84.

Treiber M. 2010. Introduction to Object Recognition. Springer-Verlag London Limited, 220 p.

Triggs B, McLauchlan P, Hartley R, and Fitzgibbon A. 1999. Bundle adjustment—a modern synthesis. In Vision Algorithms'99. 298–372.

Trucco E, and Plakas K. 2006. Video tracking: A concise survey. IEEE Journal of Oceanic Engineering, 31(2):520–529.

Tsumura T. 1986. Survey of automated guided vehicle in Japanese factory. Proc. Int'l Conf. Robotics and Automation, pp. 1329-1334.

Ulrich M, Wiedemann C, Steger C. 2009. CAD-based recognition of 3D objects in monocular images. IEEE Int. Conf. On Robotics and Automation, pp. 2090–2097.

Vacchetti L, Lepetit V, and Fua P. 2004. Stable real-time 3D tracking using online and offline information. IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, n. 10, pp. 1385-1391.

Valtkamp R, and Hagedoorn M. 1999. State of the art in shape matching. Technical Report UU-CS-1999-27, Utrecht.

Van den Heuvel, F. 1999. A line-photogrammetric mathematical model for the reconstruction of polyhedral objects. In Proceedings of SPIE, edited by V. I. Videometrics and S. F. El-Hakim, Vol. 3641, 60–71. Bellingham, WA: SPIE Publications.

Vincze M. 2001. Robust tracking of ellipses at frame rate. Pattern Recognition. 34(2): 487-498.

ViSP. 2013. ViSP: Visual servoing platform – Lagadic research platform. Retrieved from www.irisa.fr/lagadic/visp/visp.html. Accessed May 1st, 2017.

Von Gioi R, Jakubowicz J, Morel J, and Randall G. 2010. LSD: A fast line segment detector with a false detection control. IEEE Trans. Pattern Analysis and Machine Intelligence, 32 (4):722-732.

Wackrow R. 2008. *S*patial measurement with consumer grade digital cameras (Ph.D. Dissertation). Loughborough University, Leicestershire, England.

Waibel M, Beetz M, Civera J, d'Andrea R, Elfring J, Galvez-Lopez D, Häussermann K, Janssen R, Montiel J, Perzylo A, Schiessle B, Tenorth M, Zweigle O, and Van de Molengraft, M. 2011. RoboEarth – A world wide web for robots. In Robotics & Automation Magazine, IEEE, 18(2): 69-82.

Wang L, and Tsai W. 1990. Computing camera parameters using vanishing-line information from a rectangular parallelepiped. Machine Vision and Applications, 3 (3): 129– 141.

Wiedemann U, and Steger C. 2008. Recognition and tracking of 3D objects. Lect. Notes Comp. Sci., vol. 5096, pp. 132-141.

Wolf P, and Dewitt B. 2000. Elements of photogrammetry with applications in GIS. 3rd Edition. Boston, MA: McGraw-Hill.

Wolfson H, and Rigoutsos I. 1997. Geometric hashing: An overview. IEEE Trans. Computational Science Eng, 4(4):10-21.

Wu C. 2013. Towards linear-time incremental structure from motion. In 3DV, 2013.

Wu C. 2014. VisualSFM: A visual structure from motion system. Retrieved from www.ccwu.me/vsfm. Assessed May 1st. 2017.

Wu C, Agarwal S, Curless B, and Seitz S. 2011. Multicore bundle adjustment. In CVPR11.

Wuest H, Vial F, and Stricker D. 2005. Adaptive line tracking with multiple hypotheses for augmented reality, Proc. ISMAR, pp. 62-69.

Wuest H, Stricker D. 2007. Tracking of industrial objects by using cad models. Journal of Virtual Reality and Broadcasting, 4(1).

Wirola L, Laine T, and Syrjärinne J. (2010). Mass market considerations for indoor positioning and navigation. Proceedings of the 2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN). Campus Science City, ETH Zurich, Switzerland.

Yokoo K, Beauregard S, and Schneider M. 2009. Indoor relative localization with mobile short range radar. Proceedings of the Vehicular Technology Conference (VTC'09), pp. 1–5.

Yoon Y, Kosaka A, Kak A. 2008. A new Kalman-filter-based framework for fast and accurate visual tracking of rigid objects. IEEE Trans. On Robotics, 24(5):1238–1251.

Zhang Z, Deriche R, Faugeras O, Luong Q. 1995. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. Artificial Intelligence, 78:87–119.

Zumberge J, Heflin M, Jeffercon D, Watkins M, and Webb F. 1997. Precise point positioning for the efficient and robust analysis of GPS data from large networks. Journal of Geophysical Research, 102(B3), 5005-5018.

APPENDIX A

A.1 Quaternion representation of attitude

The quaternion representation of attitude has several advantages over the Euler angle and DCM representations. Mainly, they avoid singularities and they are more computationally efficient. The quaternion differential equations are linear and do not use trigonometric functions, and the small number of parameters (Farrell and Barth, 1998). The quaternion is a four-parameter representation of attitude, based on the idea that a transformation from one coordinate frame to another is performed by a single rotation about a vector μ (Titterton and Weston, 1997). A quaternion is a four-element vector:

(A.1)

$$\mathbf{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \frac{\mu_x}{\mu}\sin\left(\frac{\mu}{2}\right) \\ \frac{\mu_y}{\mu}\sin\left(\frac{\mu}{2}\right) \\ \frac{\mu_z}{\mu}\sin\left(\frac{\mu}{2}\right) \\ \cos\left(\frac{\mu}{2}\right) \end{bmatrix}$$

where μ_x, μ_y, μ_z, are components of the rotation angle vector **μ**, and $\mu = \sqrt{\mu_x^2 + \mu_y^2 + \mu_z^2}$. The quaternion should satisfy the following normality condition:

(A.2)

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$$

When this condition is not fulfilled, the normalization of the quaternion can be applied:

(A.3)

$$\hat{\mathbf{q}} = \frac{\mathbf{q}}{\sqrt{\mathbf{q}^T\mathbf{q}}}$$

The transformation between the Euler roll, pitch, and yaw angles and quaternion is:

(A.4)

$$\mathbf{q}=\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}=\begin{bmatrix} \cos\left(\frac{\varphi}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right)+\sin\left(\frac{\varphi}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right) \\ -\cos\left(\frac{\varphi}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right)+\sin\left(\frac{\varphi}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) \\ \cos\left(\frac{\varphi}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right)+\sin\left(\frac{\varphi}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right) \\ \cos\left(\frac{\varphi}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right)-\sin\left(\frac{\varphi}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) \end{bmatrix}$$

and

(A.5)

$$\begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix}=\begin{bmatrix} \tan^{-1}\left(\frac{2\left(q_2q_3+q_0q_1\right)}{1-2\left(q_1^2+q_2^2\right)}\right) \\ -\sin^{-1}\left(2\left(q_1q_3-q_0q_2\right)\right) \\ \tan^{-1}\left(\frac{2\left(q_1q_2+q_0q_3\right)}{1-2\left(q_2^2+q_3^2\right)}\right) \end{bmatrix}$$

The transformation between the quaternion and the DCM is:

(A.6)

$$\mathbf{R}_{\mathbf{M}}^{\mathbf{b}}=\begin{bmatrix} 1-2\left(q_2^2+q_3^2\right) & 2\left(q_1q_2+q_0q_3\right) & 2\left(q_1q_3-q_0q_2\right) \\ 2\left(q_1q_2-q_0q_3\right) & 1-2\left(q_1^2+q_3^2\right) & 2\left(q_2q_3+q_0q_1\right) \\ 2\left(q_1q_3+q_0q_2\right) & 2\left(q_2q_3-q_0q_1\right) & 1-2\left(q_1^2+q_2^2\right) \end{bmatrix}$$

and

(A.7)

$$\mathbf{q}=\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}=\begin{bmatrix} \frac{r_{32}-r_{23}}{2\sqrt{1+r_{11}+r_{22}+r_{33}}} \\ \frac{r_{13}-r_{31}}{2\sqrt{1+r_{11}+r_{22}+r_{33}}} \\ \frac{r_{21}-r_{12}}{2\sqrt{1+r_{11}+r_{22}+r_{33}}} \\ \frac{\sqrt{1+r_{11}+r_{22}+r_{33}}}{2} \end{bmatrix}$$

where r_{ij} are the elements of the $\mathbf{R}_{\mathbf{M}}^{\mathbf{b}}$.